PROYECTO

FIN DE CARRERA

Universidad Carlos III de Madrid

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

DISEÑO Y CODIFICACIÓN DE UN COMPLEMENTO MULTILENGUAJE INTEGRADO EN PAQUETES DE PRODUCTIVIDAD PARA FACILITAR LA TRANSFORMACIÓN DEL LENGUAJE NATURAL EN SENTENCIAS DE INTERACCIÓN CON BASES DE DATOS

AUTOR

D.CARLOS CÓCERA PÉREZ

DIRECTOR

D. ENRIQUE JIMÉNEZ DOMINGO

"El éxito no es para los que piensan que pueden hacer algo, sino para quienes lo hacen."

(Anónimo)

AGRADECIMIENTOS

En primer lugar y siendo lo más importante para mí, quiero dedicar este proyecto a mi familia que es lo que más quiero, en especial a mi madre. Quiero agradecerle lo que, cuando uno es más joven no dejas de reprocharle, y es que haya sido tan persistente y fuerte como para empujarme a, no solo terminar este proyecto, sino también a conseguir ser Ingeniero; la mitad del mérito y el esfuerzo de toda mi carrera como estudiante te corresponde mamá. Gracias por estar a mi lado y apoyarme incondicionalmente, contagiarme con tu fuerza cuando la he necesitado, con tu templanza cuando estaba tenso, con tus ánimos cuando estaba desmoralizado....eres un mar de virtudes y por todo esto te estaré agradecido toda mi vida.

También agradecerles a mis hermanas, Ana y Elena, y a mi novia que me hayan apoyado y animado incondicionalmente cuando me he enfrentado a adversidades, y particularmente a ti Ana, por ser mi segunda madre durante tantos años y enseñarme la virtud de la constancia y del trabajo.

Por último quiero agradecer a Ricardo su increíble labor, dedicación, entrega, preocupación e inmenso esfuerzo invertido en este proyecto; en atreverte a ser mi tutor sin conocerme de nada, bueno más que un tutor, un amigo, gracias por compartir conmigo tu experiencia y enriquecer la mía. Sé que no soy una persona fácil de llevar y hay que estar a veces muy pendiente, tener paciencia y apretarme para que no me relaje. Ricardo, tú tienes esas virtudes entre otras muchas que no voy a mencionar, es mejor conocerlas en persona ¡Eres un “crack”!

No voy olvidar mencionar el gran trabajo de Enrique, que se ha aventurado a continuar el trabajo de Ricardo, pese al inconveniente de incorporarse a medio proyecto y a la presión por la escasez de tiempo; muchas gracias Enrique por tu enorme esfuerzo y por no dejar en ningún momento de motivarme y darme ánimos.

Y por último agradecer a mis amigos que me hayan dado esos momentos de desconexión que todos necesitamos para recargar pilas y no olvidar a mis compañeros de trabajo, de los que tanto he aprendido y a de los que siempre conservaré un grato recuerdo. ¡GRACIAS A TODOS!

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

RESUMEN

A medida que las nuevas tecnologías evolucionan en una era basada en la informática y las telecomunicaciones, incrementa el riesgo de que las compañías no dispongan del mejor software orientado a la gestión de la información que almacenan.

Las empresas ya no se conforman con que únicamente sea un empleado experto que disponga de los conocimientos técnicos necesarios para realizar operaciones de gestión de datos y buscan facilitar dicha tarea para que, casi cualquier empleado, pueda hacerlo. Esto obliga a que los desarrolladores dediquen gran parte de sus esfuerzos en implementar interfaces de diálogo cada vez más sencillos que permitan operaciones de gestión de información sobre una base de datos, de un modo convencional, como lo haríamos usando el lenguaje cotidiano y común.

Este trabajo representa el desarrollo de un componente de aplicación integrado en paquetes de productividad Microsoft Office [1]que consiste en un traductor de Lenguaje Natural a Bases de Datos mediante una interfaz y que permite traducir consultas en lenguaje natural a sintaxis SQL que aporte y cubra estas necesidades de comunicación.

[1] http://www.office.microsoft.com/es-es/

ABSTRACT

As new technologies evolve on information and communication based era, technologies increases the risk that companies do not have the best software designed to manage the stored information.

Companies are no longer satisfied with only an expert employee with the knowledge necessary to perform data management and they seek facilitating this task for almost any employee. This requires that developers devote much of their efforts implementing ever simple user dialog interfaces that allow DataBase management operations, in a conventional way, as we would do using the common daily language.

This work represents the development of an integrated Microsoft Office productivity suite application component which consists in a translator of Natural Language on Databases languages through an interface that makes possible translating natural language queries to SQL syntax so it can solve this communication needs.

CAPÍTULO I

INTRODUCCIÓN

1.1. INTRODUCCIÓN

Este trabajo intentará servir a tres propósitos: Introducir al lector en el área de ILNBD describiendo algunas de las tareas centrales en el procesamiento del lenguaje natural, indicar la situación en esta área subrayando las facilidades y métodos típicamente implementados en este tipo de sistemas y proponer el diseño de un sistema de ILNDB en español.

El lenguaje natural es uno de los medios más poderosos de comunicación. El desarrollo de ILNBD para cualquier tipo de bases de datos ha llegado a ser muy común debido a que la facilidad en la comunicación es esencial para la aceptación del usuario. Se dice que un programa comprende el lenguaje natural si este se comporta seleccionando una acción correcta o aceptable en repuesta a la consulta. La acción tomada no necesita ser una respuesta externa. Esta puede ser simplemente la creación de alguna estructura interna de datos, o invocaciones de rutinas de acción para desarrollar la tarea deseada.

Estos sistemas pueden ser herramientas extremadamente efectivas para el acceso a bases de datos, y podrían usar ciertos lenguajes naturales (usualmente inglés). Aunque actualmente el problema de usar sin restricciones las consultas en lenguaje natural para cualquier lenguaje natural ha sido resuelto.

Sin embargo, las interfaces que solamente usan el teclado como entrada del lenguaje natural no son del todo prácticas; a largo plazo, el uso de entradas habladas y/o la integración con interfaces gráficas de usuario, parecen ser la ruta a seguir para generar ILNBD exitosas. A pesar de esto, las ILNBD basadas en comandos escritos aún son útiles en el desarrollo de pruebas para el desarrollo de nuevas técnicas para su subsecuente incorporación dentro de estos sistemas más complejos.

A pesar de que los primeros trabajos sobre interfaces de lenguaje natural surgieron en 1947, es hasta finales de los sesentas y principios de los setentas que se desarrollan las primeras

interfaces de lenguaje natural para bases de datos, debido a que las bases de datos no alcanzaron su madurez hasta la aparición del modelo relacional de Codd [1]. Uno de los proyectos más sobresalientes de los años setenta fue LUNAR [2], el cual manejaba una base de datos de estructura particular con información acerca de rocas lunares. Esta interfaz tenía una arquitectura basada en la sintaxis y predicados de primer orden, debido a lo cual era muy difícil de configurar para utilizarse con nuevos dominios. LUNAR ayudaba al usuario a formular su consulta a través de diálogos. Por otra parte está RENDEZVOUS [3] (1976-1977), un lenguaje de consultas cercano al inglés y basado en Codd que aplicaba técnicas semejantes a las de LUNAR.

A finales de los setentas aparecieron LADDER [4], PLANES [4] y PHILIQA1 [4], los cuales empleaban gramáticas semánticas, una técnica que relacionaba el procesamiento sintáctico y semántico. La interfaz PHILIQA1 se focalizó principalmente en problemas semánticos y, al igual que LADDER, podía ser usada con grandes bases de datos y diferentes manejadores, pero debía ser configurada manualmente. PLANES, por el contrario, ayudaba al usuario a formular sus consultas mediante diálogos.

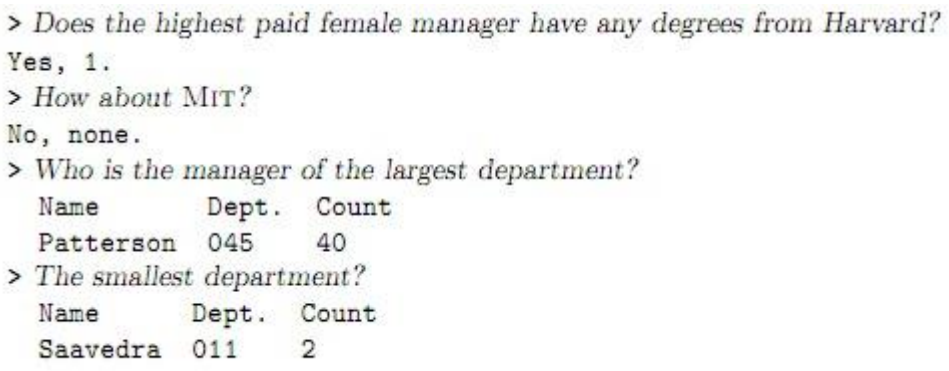
Las gramáticas semánticas manejadas por los sistemas en los años setentas dificultaban la portabilidad del dominio, por lo que los investigadores abandonaron este tipo de técnicas y durante la década de los ochentas se enfocaron en la portabilidad del dominio. A principios de los ochentas aparece CHAT-80 [5]. Este sistema traducía consultas en inglés a expresiones en Prolog, las cuales eran evaluadas sobre una base de datos de Prolog. Para realizar la traducción se utilizaban predicados lógicos. Un ejemplo del lenguaje CHAT-80:

```
>>> from nltk.sem import chat80
>>> print chat80.items
('borders', 'circle_of_lat', 'circle_of_long', 'city', ...)
```

Otras interfaces que también utilizaban predicados lógicos son NATLIN [7] y TELI [7]. NATLIN fue implementada en Prolog pero únicamente genera las consultas y, así como TELI, realiza un análisis sintáctico y semántico de la oración. A mediados de los ochentas aparecen

JANUS [8], DATALOG, LDC, TQA, EUFID [8] y TEAM [8]; y todavía algunas de las interfaces como JANUS Y EUFID utilizan la técnica de gramáticas semánticas. JANUS fue uno de los pocos sistemas en su tiempo que soportaba consultas temporales donde el usuario podía realizar consultas en tiempo presente, pasado y futuro. En lo que respecta a la portabilidad de las interfaces, TEAM establece la importancia de dividir la información dependiente del dominio de la información que pudiera ser manejada de manera general, con la finalidad de proporcionar independencia a la interfaz del dominio que se esté utilizando, aunque no realizó grandes avances al respecto debido a que la técnica de traducción utilizada resultaba sumamente dependiente del dominio. TEAM tenía tres componentes principales: un componente de adquisición, un módulo basado en diálogos y un componente para acceso a datos.

Además, presentaba los principios de la arquitectura de lenguaje de representación intermedio, ya que la consulta era convertida a una forma lógica y posteriormente a una representación formal. A finales de los ochentas emergen las primeras ILNBDs comerciales, entre las cuales se encuentra BBN´s Parlance [9]. BBN´s Parlance tenía módulos morfológicos que permitían al sistema determinar las diferentes formas de las palabras. Un ejemplo de diálogo con BBN's Parlance sería el siguiente:


> *Does the highest paid female manager have any degrees from Harvard?*
Yes, 1.
> *How about* MIT*?*
No, none.
> *Who is the manager of the largest department?*

Name	Dept.	Count
Patterson	045	40

> *The smallest department?*

Name	Dept.	Count
Saavedra	011	2

Figura 1.- Ejemplo de diálogo con interfaz BBN's Parlance

Hasta antes de la década de los noventas, la mayoría de las ILNBDs utilizaban como lenguaje natural el idioma inglés, pero a partir de esta fecha importantes investigaciones sobre

las interfaces para el idioma español comienzan a surgir. Entre éstas se encuentran ILNES [10], PASO PC-315 [11], NATLIN y SISCO.

En el Instituto Tecnológico de Monterrey Campus Cuernavaca se creó el sistema ILNES. Este sistema consta de cinco componentes principales: un diccionario de símbolos, un diccionario de sinónimos, un diccionario de datos, un modelador del contexto de la base de datos, y un intérprete de lenguaje natural. Para obtener la consulta en SQL el intérprete utilizaba patrones que representan el conocimiento de la interfaz.

NATLIN fue desarrollada en la Universidad de Essex en Inglaterra, pero no podía utilizar una base de datos comercial porque generaba la consulta en Prolog; esto la limitaba notablemente, ya que no podía probarse con datos más complejos. Por tal motivo se continuó este proyecto en México, en la Universidad de las Américas (Puebla), donde adaptaron el sistema NATLIN para la generación de la consulta en SQL.

PC315 de la Universidad Politécnica de Madrid[2]y SISCO de la Universidad Politécnica de Alicante[3] están basados en gramáticas de tipo lógico-modular.

En los años noventa se consolidan las ILNDBs con el surgimiento de MASQUE/SQL [12]. Esta interfaz está basada en la arquitectura de Lenguaje de Representación Intermedio y cuenta con un lexicón que tiene una lista de las posibles formas de las palabras que podrían ser usadas en la consulta del usuario, así como expresiones lógicas que describen el significado de cada palabra. El lenguaje intermedio que utiliza es LQL, y para obtener la consulta en SQL realiza mapeos a la base de datos y diccionarios mediante predicados lógicos. Puede manejar diferentes dominios, pero es necesario que éstos sean construidos mediante un editor de configuración que es semiautomático. También durante la década de los noventas aparecen algunas interfaces comerciales, las cuales eran un poco más sofisticadas en comparación con las interfaces de la

[2] http://www.upm.es/ Último acceso 13-05-2011
[3] http://www.eps.ua.es/ Último acceso 13-05-2011

década anterior, pero todavía presentaban notables desventajas. A continuación se presentan las más importantes:

SYSTEMX [13]: Este sistema presenta un diseño modular, ya que posee un módulo para el tratamiento de la consulta en lenguaje natural y otro para el análisis de la base de datos.

LOQUI [14]: Sistema desarrollado en Prolog por BIM, obtenía una consulta lógica y reglas expresadas en Prolog. La consulta lógica era tratada por el interpretador de Prolog y la información de la base de datos era tratada como hechos de Prolog.

INTELLECT [15]: Comercializado por IBM, era configurado para interactuar con la base de datos a través de un sistema experto en C, de esta manera se podía agregar un cierto de razonamiento entre la ILNDB y la base de datos.

NATURAL LANGUAGE [16]: Sistema que tenía diccionarios que contenían una lista de las palabras más comunes. Los diccionarios podían ser administrados sólo por las personas que configuraban el sistema incluyendo la terminología del dominio específico.

Q&A [17]: Sistema basado en menús, donde el usuario no podía escribir directamente sus consultas, por lo que éstas tenían que ser construidas escogiendo posibles palabras o frases desde menús.

SQ-HAL [18]: Sistema que traduce consultas en lenguaje natural a consultas simples en SQL y está desarrollado en Perl. Posee un analizador sintáctico descendente recursivo y define una gramática propia (reglas de producción).

EDITE [19]: Surge entre 1997 y 1999 como uno de los primeros ILNBDs multilingües. EDITE estaba desarrollado siguiendo la arquitectura de Lenguaje de Representación Intermedio, utilizando LIL como lenguaje intermedio.

TAMIC-P [20]: Sistema que maneja el idioma Coreano utilizando OQL como lenguaje intermedio y patrones sintácticos para realizar la traducción, los cuales se construyen manualmente. Para realizar la traducción se auxilia de mapeos a las bases de datos para tener una consulta previa en lenguaje intermedio OQL y CPL. Todos los proyectos anteriormente mencionados enfocan su atención en el análisis léxico, sintáctico y semántico de la consulta en lenguaje natural; y en lo que respecta a la traducción de la oración de entrada, en una representación formal que pueda ser manipulada por la computadora con éxito. Las técnicas tradicionales que incorporan desde reconocimiento de patrones hasta predicados lógicos limitan a la interfaz a dominios y a bases de datos predefinidas por el administrador de la ILNBD. Por tal motivo se han comenzado a realizar nuevos proyectos que incorporan las ventajas de las técnicas utilizadas por los sistemas convencionales, pero con nuevas propuestas. A continuación se describe una de ellas.

PRECISE [21]: Fue desarrollado en la Universidad de Washington[4], y para realizar la traducción de una consulta en lenguaje natural, primero determina si es posible tratarla semánticamente. Para saber esto, mapea los elementos léxicos de la oración a los elementos de la base de datos a través de restricciones semánticas impuestas, reduciendo el problema a una comparación de grafos, el cual es resuelto con un algoritmo de flujo máximo. La desventaja principal es que el grafo semántico es diseñado manualmente por el usuario a través de un editor.

En la Universidad de California [5]se está desarrollando una ILNBD que, para realizar la traducción, utiliza un modelado semántico que consiste en un grafo semántico de la base de datos, el cual representa la información almacenada de ésta. La interfaz acepta del usuario la consulta en lenguaje natural y extrae la información necesaria.

[4] http://www.washington.edu/ Último acceso 13-05-2011
[5] http://www.ucla.edu/ Último acceso 13-05-2011

Figura 2. Ejemplo de un análisis erróneo corregido por PRECISE.

El proceso de extracción es realizado usando palabras clave obtenidas del grafo semántico de la base de datos. Pero debido a que las palabras clave pueden tener diferentes significados sin un dominio dado, es necesario evitar ambigüedad de significados de las palabras clave usando aproximaciones estadísticas que involucran la comparación de vectores de n-gramas. Esta interfaz permite utilizar diferentes dominios, pero está limitada a oraciones de entradas cortas y simples, además de depender de una ontología limitada que soporta al dominio.

Para finalizar mencionar que, en Corea, se ha desarrollado una interfaz que utiliza patrones léxico-semánticos y gramáticas multinivel.

Por todo lo comentado se propone lo siguiente: dada una estructura de datos que contiene una oración en español, junto con las categorías sintácticas de cada palabra de la oración e información adicional, traducir la oración a una expresión en SQL semánticamente equivalente.

1.2. PROBLEMÁTICA

Es cada vez más grande el número de usuarios de computadoras que no tienen un conocimiento profundo de los comandos u órdenes internas requeridas para un mejor aprovechamiento de los recursos de información que provee un sistema computacional. La traducción de lenguas naturales es un proceso complejo que requiere una vasta cantidad de información léxica y de conocimiento del mundo que los usuarios no pueden controlar.

Los sistemas de representación del conocimiento actuales permiten una aplicación del conocimiento humano para muchas tareas complejas, entre ellas la traducción. Tratar de reproducir de forma artificial las estructuras de conocimiento que un traductor humano emplea nos puede llevar, cuando menos, a un mejor entendimiento de estas estructuras de conocimiento y de los procesos que tienen lugar en la mente humana por lo que respecta a la comprensión del lenguaje.

Hasta la fecha, las interfaces de lenguaje natural para bases de datos no garantizan la traducción satisfactoria de la consulta en lenguaje natural o la obtención de una respuesta confiable. Además, es importante mencionar que en lo que respecta a la independencia del dominio todavía hay mucho trabajo que realizar, ya que el proceso de configuración a diferentes dominios y bases de datos aún es una meta lejana, debido a que depende de los diccionarios, los mecanismos de traducción y el análisis de sinónimos, antónimos y preposiciones entre otras cosas. En todos los trabajos revisados se observa que la portabilidad del dominio es un problema que está lejos de ser resuelto, esto se debe principalmente a los siguientes factores:

- El lenguaje natural siempre será más expresivo que un lenguaje formal como SQL. Además, para realizar una consulta en lenguaje natural a una base de datos es necesario conocer si la consulta es factible de ser contestada.
- La mayoría de las interfaces existentes basan su funcionamiento en la búsqueda de palabras clave dentro de la oración de entrada que están relacionadas con una acción a

realizar. La desventaja principal de utilizar palabras clave, es que estas se deben crear y configurar para cada dominio por un administrador de la base de datos o por alguien que conozca la estructura de la base de datos. Es importante señalar que, al sólo evaluar las palabras clave que aparecen dentro de la consulta, se puede pasar por alto información importante que sea necesaria para una mejor interpretación de la consulta.

1.3. SOLUCIÓN A LA PROBLEMÁTICA

En función de los problemas identificados en el apartado anterior, se pueden deducir dos posibles soluciones:

- Capacitación a los usuarios.
- Construir sistemas que comprendan instrucciones en lenguajes de más alto nivel, como el lenguaje natural o de uso cotidiano.

La primera opción tiene desventajas claras como el costo en tiempo y dinero que implica la capacitación de personal; además, una capacitación a usuarios comunes no garantiza que estos se conviertan en expertos.

La segunda opción, un tanto más viable para el usuario, representa un reto para las ramas del desarrollo de sistemas, pues no existen modelos completos para la comprensión e interpretación del lenguaje natural, sin embargo, en un dominio restringido, es posible la creación de sistemas muy poderosos que resuelvan efectivamente los problemas de su ámbito. Confrontando el reto anteriormente mencionado, se desarrolla este Proyecto de Fin de Carrera, un sistema que interpreta consultas en lenguaje natural a un sistema de información.

1.3.1. CONCLUSIONES

Es evidente el gran esfuerzo dedicado por los desarrolladores a conseguir traducciones del lenguaje natural a lenguajes de manipulación de datos lo más fieles posibles al lenguaje origen, con el fin de disminuir la pérdida de contenido semántico y corregir problemas de ambigüedad, separación entre palabras, acentos extranjeros, regionalismos, etc., que un sistema artificial no puede recrear ni detectar por sí mismo. Se ve una clara evolución desde sistemas sencillos de traducción hacia implementaciones de lenguajes de traducción con módulos intermedios y diccionarios lo más completos posibles, hasta las generación de estructuras mucho más complejas de análisis morfo sintáctico como los árboles de derivación gramatical y grafos.

1.4. ESTRUCTURA DE LA MEMORIA

A continuación, se explicará con brevedad la estructura que seguirá el desarrollo de este proyecto.

La primera parte consta de los Capítulos I y II. El Capítulo I está formado por un apartado dedicado a una breve introducción que describe qué se pretende hacer, otro que resuelve qué problemas se prevén y los tres últimos apartados que delimitan la solución propuesta, definen el marco de trabajo y plantean los objetivos, respectivamente. El Capítulo II tiene como objetivo introducir al lector en un ámbito teórico de desarrollo del proyecto y hacer un análisis de la situación actual. Finalmente describe las conclusiones al análisis que, además, permitirán establecer una base sólida sobre la que fomentar el desarrollo de este proyecto.

El Capítulo III recoge los apartados 1, 2, y 3. Estos abordan las características y funcionalidad del sistema a desarrollar, qué herramientas y tecnologías se van a emplear para su consecución y la metodología de desarrollo que se ha decidido aplicar. Todo en conjunto será lo que hará que el proyecto NQPL sea posible. EL resto de apartados, 4, 5 y 6, recordarán la línea de investigación y nuevos interrogantes que se dejarán abiertos tras este proyecto, además de los resultados obtenidos.

El último Capítulo está constituido por los anexos. Cada uno de ellos será el resultado de aplicar la metodología elegida a las fases del proyecto. Los apéndices recogidos son:

- Estudio de Viabilidad del Sistema (EVS).
- Análisis del Sistema de Información (ASI).
- Gestión del Proyecto (GP).
- Manual de Usuario (MU).

Para finalizar, se incluirá un apartado de “BIBLIOGRAFÍA”, en el que se citarán las fuentes utilizadas para llevar a cabo la investigación y ejecución de este Proyecto de Fin de Carrera.

CAPÍTULO II

ESTADO DEL ARTE

2.1. EL PROCESAMIENTO DEL LENGUAJE NATURAL

Escribir una carta, leer un periódico, ver el noticiero, tener una conversación, el diario lenguaje hablado y escrito de tales actividades es llamado Lenguaje Natural (LN) para distinguir este de otros lenguajes tales como los lenguajes de programación. Por más de treinta años, los investigadores han estudiado como las computadoras pueden ser programadas para entender y generar texto escrito y expresiones habladas. El área de estudio ha sido llamada Procesamiento del Lenguaje Natural[22] (PLN) o lingüística computacional.

En estos días, la investigación en el área del PLN es desarrollado en muchas universidades y en laboratorios de investigación de grandes compañías, además de existir un pequeño grupo de productos comerciales de PLN El interés comercial existe, y la paga por un sistema exitoso de PLN es potencialmente enorme, particularmente en aplicaciones como interfaces en lenguaje natural, sistemas de traducción, y reconocedores del habla. Las Interfaces en Lenguaje Natural (ILN) permiten a la gente comunicarse con máquinas, en un lenguaje natural como el español o el inglés.

Las ILN tienen un gran problema que es común a las máquinas traductoras o algún otro sistema de entendimiento de lenguaje natural. Estas incluyen varias formas de ambigüedad estructural, el uso de ambigüedad léxica, anáforas, incompletitud, redundancia y algunos otros.

Algunos de estos problemas se reducen al limitar el dominio de las interfaces de bases de datos a un subconjunto del lenguaje natural. El subconjunto del Lenguaje Natural debe tener un poder de expresividad suficiente para evitar la ambigüedad y redundancia en los resultados. Tradicionalmente la mayor parte de las ILNBD han seguido un paradigma particular, donde las consultas en lenguaje natural se traducen directamente a consultas formales a bases de datos (algún lenguaje de consulta formal). Estas consultas entonces se procesan por el Sistema Gestor de Bases de Datos, SGBD.

Los datos requeridos generarán una versión en lenguaje natural. La mayoría de los sistemas asumen que el lenguaje natural presentado en la entrada del sistema es al menos en cierto grado composicional, es decir que el significado de toda la oración se puede representar en función del significado de las partes que la componen. Dada una frase de entrada en lenguaje natural, el sistema crea algún tipo de estructura para la oración que describe cómo se combinan los componentes del lenguaje. Esta estructura refleja la visión del sistema de lenguaje natural.

Algunos sistemas trabajan con la estructura de la frase, mientras que otros más producen una representación lógica de la frase. El sistema necesita información sobre el tipo de palabras y frases que pueden ocurrir en el lenguaje humano que este esperaría procesar, cómo las palabras pueden combinarse juntas y cómo asignarle una estructura a la combinación de las palabras y frases. Esta información puede contener restricciones morfológicas, sintácticas y de vocabulario, conocida como conocimiento sintáctico. El conocimiento semántico en general describe, cómo se puede construir la consulta a la base de datos, es decir describe cómo las partes de la frase de entrada se pueden mapear en conceptos conocidos para la base de datos y cómo las partes mapeadas se pueden combinar juntos para crear resultados. Este es la cuestión que se manejará en este proyecto de fin de carrera.

Objetivo del capítulo

El propósito fundamental de este capítulo consistirá en realizar una breve introducción al Procesamiento del Lenguaje Natural, para poner en conocimiento del lector la situación de demanda de software actual. Se analizará además el nivel de necesidad de herramientas automatizadas que faciliten la interacción usuario-sistema y que deberán ser intuitivas y amigables y que principalmente aprovechen y reconozcan de un lenguaje natural común a todas ellas.

Otro de los objetivos del capítulo será concienciar al lector acerca de la problemática existente a la hora de unificar lenguajes computacionales y hacerle comprender el motivo, necesidad y finalidad de la solución que se va a proponer.

El objetivo final será el de proporcionar, a cualquier tipo de usuario con cualquier nivel de conocimiento del lenguaje SQL, una herramienta integrada como componente de MS Office que proporcione una interacción sólida y eficaz con sistemas de información más vanguardistas basados en SQL y que, sobre todo, sea intuitiva, sencilla, eficiente y portable.

2.1.1. CONTEXTO HISTÓRICO: LAS INDUSTRIAS DE LA LENGUA

Los orígenes del PLN constatan en los años que sucedieron a la Segunda Guerra Mundial. Por aquel entonces, únicamente Estados Unidos [6]poseía la tecnología suficiente como para llevar a cabo investigaciones de gran nivel, siendo el referente en la evolución desde sistemas toscos de traducción hasta la generación y diseño de algoritmos y software de gran potencia y resolución.

"Todo sistema de Procesamiento del Lenguaje Natural intenta simular un comportamiento lingüístico humano"

En los comienzos de los años 70, ya en Europa, el PLN surgió como piedra angular de los procesos automatizados computacionales previos a la aparición Ingeniería de la Inteligencia Artificial. Durante este periodo, empezaron a desarrollarse los primeros programas ensambladores los cuales generaban instrucciones en código máquina (*figura 4*) que se correspondían con una única instrucción/macroinstrucción de código fuente (*figura 5*)

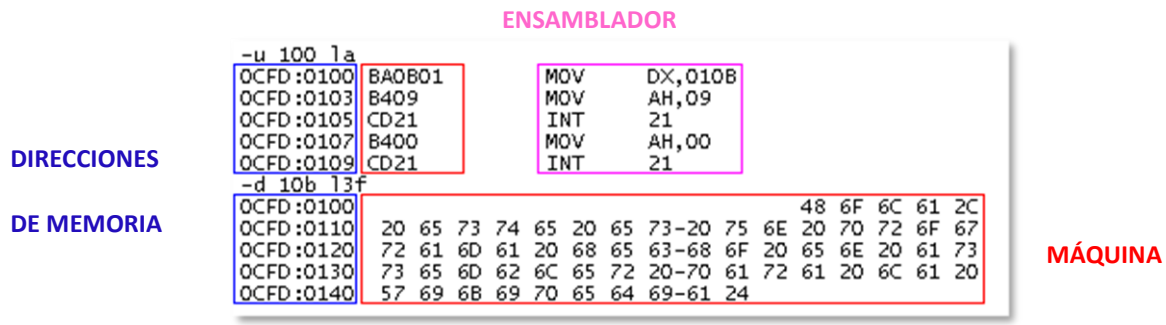


Figura 3. Código máquina.

La complejidad de estos lenguajes provocaban enormes demoras de tiempo y exceso de uso de recursos para poder realizar tareas muy sencillas tales como operaciones del tipo +,-, etc. Para agilizar y solucionar el proceso y coste de codificación, se comenzaron a diseñar programas

[6] http://www.usa.gov/Espanol/. Último acceso 01-06-2011

ensambladores capaces de generar una cantidad variable de instrucciones (macroinstrucción) en código máquina por cada instrucción del programa fuente. Así pues, el programador podía ejecutar la macroinstrucción *"Cerrar archivo"*, *"Reservar memoria"* y *"Grabar registro en memoria"*.

Figura 4. Interacción usuario-máquina mediante ensamblador.

Como consecuencia del uso de macroinstrucciones y reglas mnemotécnicas se comenzaron a desarrollar lenguajes de alto nivel pero que en principio estaban orientados a un tipo determinado de proceso, tales como problemas matemáticos, procesamiento de archivos, con un ámbito de desarrollo muy particularizado aunque posteriormente se planificarían ampliaciones de alcance. Surge así la necesidad de un cambio.

2.1.2. PLN. DEFINICIÓN.

Una vieja idea que aún no ha podido llevarse a cabo por sistemas de información expertos e ingeniería artificial es la de poder generar un lenguaje natural que los ordenadores sepan interpretar como si fuera código máquina o binario.

"Es una disciplina que relaciona directamente la informática con la lingüística"

El concepto de Procesamiento de Lenguaje Natural *(en inglés, Natural Language Processing)* al contrario de lo que se pueda pensar, no surgió como necesidad de representación y abstracción de objetos existentes en un entorno real. Surgió para afrontar la necesidad de desarrollar programas que computacionalmente, simularan una comunicación natural entre dos personas mediante sentencias, métodos y procedimientos.

Podría afirmarse que lo hicieron para permitir que la entrada de un proceso automatizado fuera el lenguaje humano.

A raíz de esta definición, surge el concepto de Lenguaje de Alto Nivel. Los lenguajes incluidos en esta categoría serían lo más cercano al lenguaje humano, independientes de la máquina, y que permiten a los usuarios olvidarse del procesamiento interno de información. Pero sobre todo, permiten la portabilidad de dicho lenguaje entre una máquina a otra, ya que lo único que requiere es la existencia un medio traductor común que lo interprete a código máquina o ensamblador.

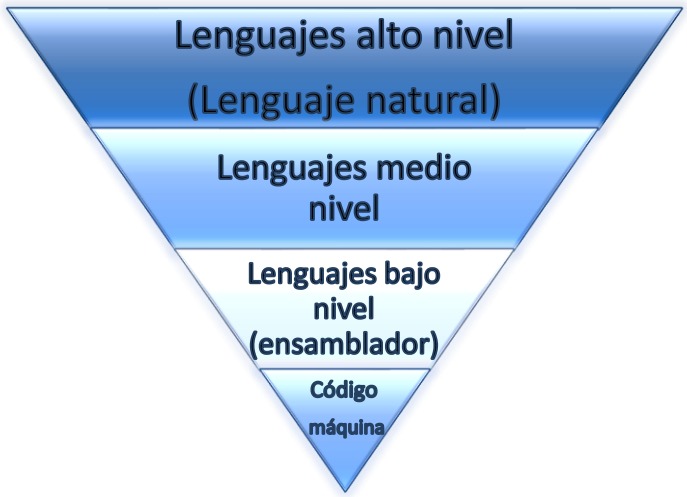


Figura 5. Niveles del lenguaje.

Con esto, los programadores disponían del anclaje necesario para poder empezar a interactuar con la máquina de una forma más cercana a cómo lo harían con otras personas y, así, iniciar su andadura por el mundo del desarrollo de aplicaciones de traducción automática del lenguaje natural.

Las principales ventajas de estos lenguajes de alto nivel frente al resto de lenguajes más cercanos a la máquina que al ser humano son:

- Son lenguajes más fáciles de aprender que los ensambladores.
- Se pueden codificar a mayor velocidad por ser sintácticamente más familiares.
- Mejoran la calidad y facilidad de documentación.

- Presentan un alto mantenimiento.
- Portabilidad a distintos tipos de máquina.

Sin embargo y paralelamente se comenzaron a detectar ciertas ambigüedades en el uso de estos lenguajes:

- A nivel léxico una palabra en un lenguaje puede tener diferentes interpretaciones, lo cual limita la traducción al contexto de la misma, complicando aún más si cabe el proceso de interpretación y traducción.
- A nivel semántico hay que conocer la estructura completa del árbol sintáctico, es decir, qué significado tiene una palabra en la estructura de la oración a la que pertenece.
- A nivel pragmático, una oración puede tener diferentes sentidos a pesar de que sintácticamente conserve la misma estructura (sarcasmo, ironía, etc). Estos problemas son los que en este proyecto se van a abordar. Se implementará una aplicación recibirá un lenguaje de entrada y lo traducirá para que la máquina lo reconozca, efectúe las instrucciones que tenga implícita dicha instrucción y finalmente se ejecuten las operaciones vinculadas a la misma.

2.1.3. EL PROCESAMIENTO DEL LENGUAJE NATURAL. ENFOQUE PRÁCTICO

El objetivo principal del proyecto será el de proporcionar una herramienta que permita una Traducción Automática de Sentencias (TAS) del lenguaje natural con el fin de interactuar con un sistema de almacenamiento de información. Como su nombre indica, la TAS se encargará de traducir textos de un idioma a otro de forma automatizada y resolutiva, conservando los aspectos semánticos. Esto es lo que llamaremos a partir de ahora Procesamiento del Lenguaje Natural.

Teniendo en cuenta los niveles del lenguaje que se han mencionado en el apartado anterior, se podrían distinguir varios sistemas de traducción automática aplicables al PLN:

- **Traducción directa**: Traducción sencilla y poco útil palabra a palabra.

- **Traducción basada en transferencia**: Sistema más avanzado que analiza el conjunto morfosintáctico en lugar de partes de frases. Es un sistema caro.
- **Interlingua:** Desarrolla lenguajes abstractos no basados en lenguajes reales lo cual permitiría por ejemplo traducir entre dos idiomas.
- **Memorias de traducción:** Requiere de la intervención del usuario para revolver ambigüedades.

En nuestro caso particular, la alternativa que más se ajustan a la necesidad de este proyecto sería un modelo intermedio entre traducción directa y traducción basada en transferencia. Podríamos aproximarnos a la interlingua pero, como se analizará más adelante, es un método de traducción costoso que requiere una cantidad muy elevada de recursos de los que no disponemos. Y para llevar a cabo esta traducción se requiere de una interacción usuario-máquina que implica la implementación de una interfaz de traducción del lenguaje natural (ILN) que permita dicha comunicación

2.1.4. LAS INTERFACES EN LENGUAGE NATURAL (ILN)

Las interfaces en lenguaje natural son frecuentemente propuestas como una solución a los problemas de “hostilidad” presente en muchas interfaces de sistemas computacionales. Como un inicio, una interfaz en lenguaje natural se define como un medio que permite al usuario controlar el sistema por órdenes en algún lenguaje tal como el español o el inglés o mediante consultas. A veces la salida a la consulta requerida por el usuario también estará en lenguaje natural.

Comúnmente, las ILNBD [23] (**I**nterfaces de traducción de **L**enguaje **N**atural a un lenguaje de interacción con **B**ases de **D**atos) sólo aceptan comandos a través del teclado, en vez de comandos de entrada hablados. También, tales interfaces, típicamente manejan una entrada restringida, relacionada con el dominio de la aplicación en uso, y además, sólo un subconjunto del lenguaje utilizado. Hay dos grandes categorías de alternativas para ILNBD:

Interfaces de comandos en línea: El usuario teclea comandos o consultas en un lenguaje artificial que generalmente es muy conciso.

Interfaces gráficas: El usuario da sus entradas por selección de menús, o mediante el uso de iconos, ventanas de ayuda, etc.

En este proyecto se va a implementar una mezcla de ambos tipos, aportando las ventajas de cada una de ellos para hacer sentir al usuario que realmente es él el que interactúa con la base de datos y no la propia interfaz y difuminar así la brecha existente entre usuarios novatos y expertos. En el siguiente apartado se detallará más a fondo las ventajas y desventajas de estos dos tipos de interfaces.

2.1.5. VENTAJAS Y DESVENTAJAS DE LAS ILNBD

Las interfaces también pueden aceptar una combinación de gráficas y comandos en línea. Las ventajas sobre otros tipos de interfaces que generalmente son atribuidas a las ILN incluyen:

- **Expresividad***:* Si puede pensar en una manera para expresar una orden o consulta en español, esta puede introducirse en una ILN.
- **No requiere de aprendizaje**: La gente puede consultar al sistema en su mismo lenguaje por lo que no requiere del aprendizaje de un nuevo idioma.
- **Naturalidad:** El lenguaje natural es "común" y por consiguiente es una forma cómoda de expresión para todo el mundo.

Por el contrario, existen varias desventajas entre las que podemos destacar las siguientes:

- **Lenguaje limitado:** Comandos o consultas en español pueden tomar mucho más tiempo y más líneas de entrada que líneas de comandos formales equivalentes o selecciones basadas en menús.
- **Restricciones de la cobertura:** Las ILN no pueden manejar todo tipo de entradas en lenguaje natural, aún aquellas relevantes a su dominio del discurso, se enfrenta al usuario con la tarea de aprendizaje acerca de lo que el sistema puede y no puede procesar, normalmente esto es por prueba y error.
- **Los usuarios asumen inteligencia:** Los usuarios de las ILN se confunden a menudo por la habilidad del sistema para procesar lenguaje natural, y asumen que el sistema es

inteligente, que tiene sentido común o que puede deducir hechos que parecen simples, mientras que la realidad es que la mayoría de ILN no tiene habilidades del razonamiento. Este problema no existe en lenguajes de consulta formales, interfaces basada en formularios o menús e interfaces gráficas, donde las capacidades del sistema son más obvias al usuario.

- **Medio inapropiado:** Se ha argumentado que el lenguaje natural no es un medio apropiado para comunicarse con un sistema de computadora. Se señala que el lenguaje natural es demasiado redundante o demasiado ambiguo para la interacción humano-computadora.
- **Consultas largas:** los usuarios de ILN tienen que teclear preguntas largas, mientras en interfaces basadas en formularios sólo algunos campos se tienen que rellenar, y en interfaces gráficas la mayor parte del trabajo se puede hacer por medio del uso del ratón.
- **Consultas ambiguas:** Las consultas en lenguaje natural a menudo son ambiguas, mientras que las basadas en formularios, o las consultas gráficas nunca tienen significados múltiples
- **Configuración tediosa:** Las ILN normalmente requieren fases de configuraciones tediosas y largas antes de que se puedan usar. En contraste la mayoría de los sistemas de bases de datos comerciales han sido desarrollados en base de intérpretes de lenguajes de consulta formales, y se automatiza la fase de configuración considerablemente.

2.1.6. FACTORES IMPORTANTES EN EL DESEMPEÑO DE LAS ILN

Entre los factores que incluyen en el desempeño y la utilidad de las ILN generalmente se incluyen:

- **Tipo de usuario:** Las ILN generalmente satisfacen mejor a usuarios novatos o usuarios casuales en contra de los usuarios expertos o más frecuentes del sistema. Un usuario experto o frecuente puede preferir el aprendizaje un lenguaje formal, debido a que son muy concretos. En cambio, sería más adecuado para un usuario principiante o poco

frecuente ingresar una entrada en lenguaje natural que formular la línea de comandos correcta en un lenguaje formal.

- **Hardware de interface:** Para que funcionen las interfaces gráficas requieren comunicación de alta-velocidad entre el procesador y la pantalla, además el computo del lenguaje natural también requiere recursos del sistema considerables. Si éstos no están disponibles, otros tipos de interfaces son preferibles.
- **Dominio de la Aplicación:** Si la aplicación sólo puede aceptar unos cuantos comandos con unos pocos parámetros del dominio seleccionado, el poder expresivo del lenguaje natural no es usado efectivamente, y una interfaz de comandos en línea o (más probable) una interfaz gráfica sería preferible. El más grande y más complejo de los medios proporcionados por la aplicación es probablemente el poder expresivo del lenguaje natural.
- **Medio de Entrada:** Si una ILN acepta lenguaje hablado en lugar de teclear la entrada, se supera el problema de escritura grande, ya que es mucho más rápido hablar que escribir. Desgraciadamente, las capacidades actuales de reconocimiento del lenguaje hablado presentes son más lentas pues detrás de ellas están los procesos del lenguaje natural escrito.
- **Grado de cobertura:** Claramente, es deseable que una ILN sea aplicable con distintos tipos de lenguajes naturales como sean posibles. Seleccionar la cobertura de un lenguaje ya es en sí una tarea difícil, sin embargo, esto depende del vocabulario, la complejidad sintáctica, entradas que contienen anáforas, elipsis, y otros fenómenos lingüísticos del diálogo. Asumiendo que la meta de cobertura completa del lenguaje es un proceso largo, se necesita determinar primero los métodos adecuados de cobertura de los dominios particulares de la aplicación para cada tarea asignada del sistema.
- **Combinación con otros tipos de entrada*:*** Sería posible construir interfaces que combinan lenguaje natural y otros tipos de interface de manera que retienen las mejores características de ambos, mientras se reduce el impacto de sus rasgos negativos.

Ahora que se tiene bien claro el concepto de PLN y el usuario ya ha entendido qué modelo de interacción con la máquina va a utilizar, se plantea la siguiente cuestión,

¿De qué tecnologías y herramientas disponemos para conseguir interactuar con un sistema de información mediante PLN y satisfacer este desempeño?

2.2. SISTEMAS DE GESTIÓN DE INFORMACIÓN

Los sistemas de gestión de información surgen como respuesta de la necesidad de las personas de gestionar grandes cantidades de información, almacenada en ficheros, de manera rápida eficaz y fiable.

No se conoce a ciencia cierta en qué momento dejó de almacenarse dicha información en ficheros y comenzaron a diseñarse estructuras capaces de organizar datos de forma consistente y aplicaciones que permitieran la interacción del usuario con dichos datos. De hecho, hoy en día se sigue almacenando información en estructuras de tipo fichero aunque en menor medida que en Bases de Datos.

En este apartado se realizará un repaso conceptual y funcional a estas estructuras para una mejor comprensión por parte del lector a las mismas.

2.2.1. CONCEPTO DE BASE DE DATOS

Una base de datos es un almacén de información que permite administrar información de manera ordenada y que facilita el manejo de su contenido. Podría definirse como un conjunto de cadenas de caracteres organizado.

¿De qué modo se manipula esta información?

A través de aplicaciones diseñadas específicamente para realizar la tarea llamadas Sistemas Gestores de Bases de Datos que se definen como una colección de datos interrelacionados y un

conjunto de programas para acceder a dichos datos. Estos SGBD son los encargados de realizar las siguientes operaciones sobre la base de datos:

Definir: Se especifican las estructuras y restricciones que los datos almacenados tendrán para facilitar las tareas de manipulación y conservación de los mismos.

Construir: Es el proceso de almacenaje de los datos sobre algún tipo de medio de almacenamiento.

Manipular: Incluye acciones sobre la base de datos tales como actualización, modificación (insertar, borrar datos.) o consulta de alguno de sus elementos.

Conservar: Tratan de conservar la integridad y consistencia de los datos para evitar futuros problemas y tener que volver a definir y construir la estructura de acceso a los mismos.

Una de las intenciones de este proyecto es ejecutar las acciones típicas de un SGDB definidas anteriormente

2.2.2. ORÍGEN DE LOS SISTEMAS GESTORES DE BASES DE DATOS

Las técnicas del almacenamiento de datos han evolucionado mucho a lo largo de los últimos 50 años. El procesamiento de datos supuso un trampolín para el crecimiento de los computadores, como ocurriera en los primeros días de los computadores comerciales. Este procesamiento de datos estuvo muy estrechamente relacionado con la automatización de las tareas debido a que el manejo cantidades de información de tal magnitud era insostenible desde un punto de vista humano.

No es sino a principios de los años 90 cuando estos sistemas eficaces, como son SQL [24], Oracle, Access, realmente se han empezado a instaurar y a consolidad su uso en las empresas y a conseguir un ritmo de desarrollo y de evolución que está permitiendo que se puedan manejar cantidades de información pantagruélicas de manera sencilla, eficiente y consistente.

2.2.3. OBJETIVO DEL DISEÑO DE BASES DE DATOS

El mundo de los datos son las cadenas de caracteres a las que nos referíamos anteriormente, y que carecen de significado por sí mismos si no disponemos de los medios necesarios para recorrer el camino inverso, extrayéndolos al mundo real con ayuda del lenguaje de manipulación, que nos permitirá recuperarlos y reincorporarlos a su contenido semántico, proporcionando la información que necesita al usuario.

Los objetivos del diseño de Bases de Datos son:

- Aportar conceptos para el diseño de BD.
- Profundizar en el modelo E/R de Chen como soporte conceptual.
- Analizar el ciclo de diseño de una BD.
- Describir la funcionalidad de los módulos o subsistemas más significativos.

2.2.4. MODELOS DE DATOS ACTUALES

CLASIFICACIÓN

En la actualidad, los SGBD según su modelo de estructura y acceso a los datos pueden clasificarse en:

- El **Modelo Relacional** se basa en el concepto matemático denominado “relación", que gráficamente se puede representar como una tabla con columnas y filas y se especifica qué datos se han de obtener.
- En el **Modelo en Red** los datos se representan como colecciones de registros las relaciones entre los datos se representan mediante conjuntos.
- El **Modelo Jerárquico** es un tipo de modelo de red con algunas restricciones: cada nodo puede tener un solo padre.
- El **Modelo Orientado a Objetos** define una base de datos en términos de objetos, sus propiedades y sus operaciones.

La mayoría de los SGBD comerciales actuales están basados en el modelo relacional, mientras que los sistemas más antiguos estaban basados en el modelo de red o el modelo jerárquico. A continuación se muestra un diagrama donde puede apreciarse el proceso de diseño de una base de datos.

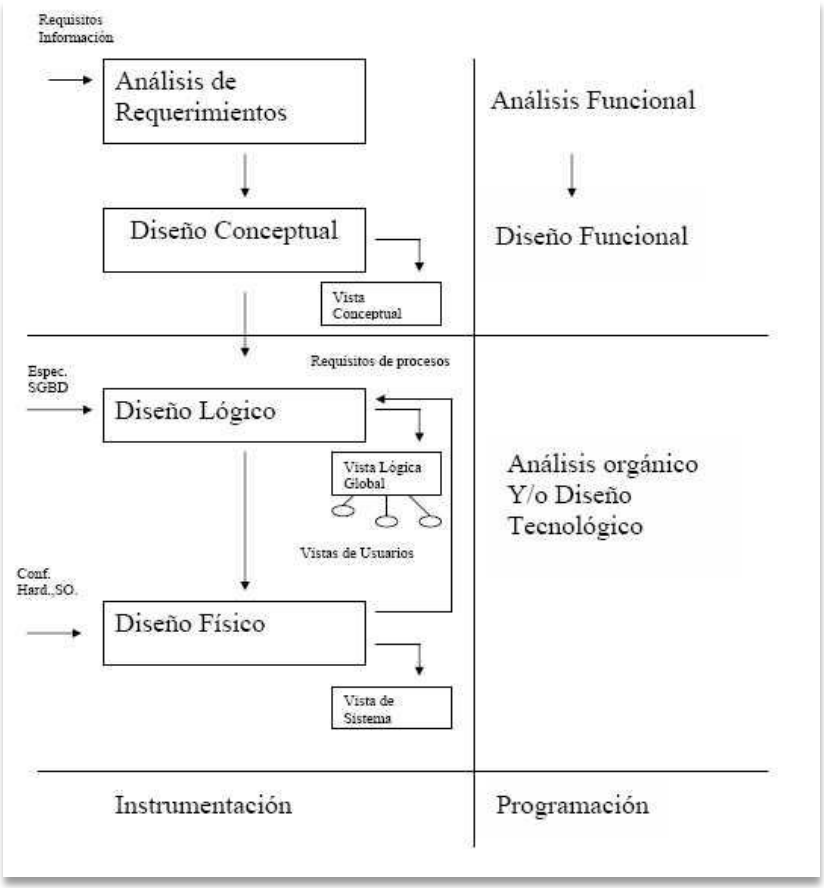


Figura 6. Esquema de las fases de diseño de una base de datos.

DISEÑO DEL MODELO CONCEPTUAL

El diseño conceptual especifica principalmente los componentes estáticos de la base de datos, incluyendo las estructuras y las restricciones estáticas. Es un componente independiente de la implementación, mientras que es dependiente de los desarrolladores y diseñadores.

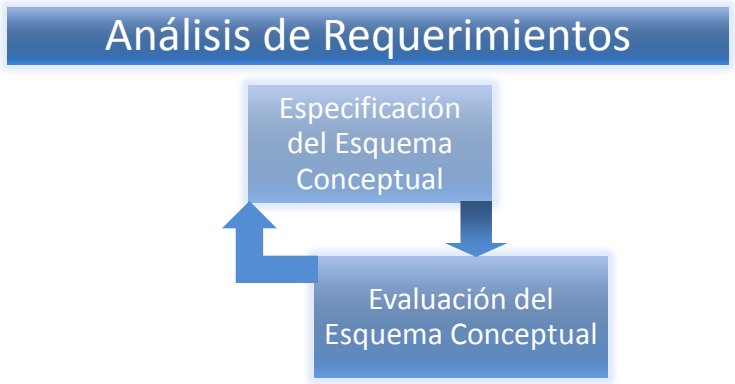


Figura 7. Esquema diseño conceptual.

Los elementos fundamentales de la fase del diseño conceptual son:

- Análisis de Requerimientos (AR).
- Especificación del Esquema Conceptual (EEC).
- Evaluación del Esquema Conceptual (EvEC).

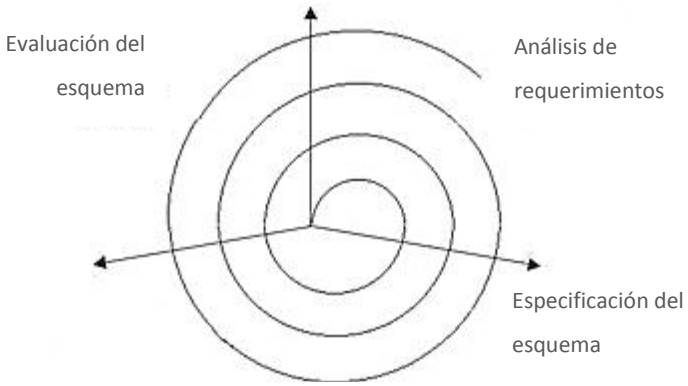


Figura 8. Proceso en espiral de diseño conceptual.

DISEÑO LÓGICO

El objetivo del diseño lógico es transformar el diseño conceptual obtenido adaptándolo al SGBD que se vaya a utilizar.

DISEÑO FÍSICO

El paso inicial del diseño de la base de datos física es la traducción del modelo de datos lógico. La finalidad del Diseño Físico es encontrar una estructura interna que soporte la estructura conceptual y los objetivos del diseño lógico con la máxima eficiencia de los recursos máquina:

- Reducir los tiempos de respuesta.
- Proporcionar un alto grado de seguridad.
- No tener que reorganizar los datos.
- Minimizar el espacio de almacenamiento utilizado.
- Optimizar el uso de recursos.

2.2.5. LENGUAJE DE CONSULTA ESTRUCTURADO SQL

El lenguaje SQL (Structured Query Language) es un lenguaje de consultas estructurado en base de datos muy difundido. SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos.

INTRODUCCIÓN AL LENGUAJE SQL

Como su propio nombre indica, SQL es un lenguaje computacional que se puede utilizar para interaccionar con una base de datos y, más concretamente, con un tipo específico llamado base de datos relacional. SQL es a la vez un lenguaje fácil de aprender y una herramienta completa para gestionar datos, propiedad por la cual está tan difundido. Las peticiones sobre los datos se expresan mediante sentencias, que deben escribirse de acuerdo con unas reglas sintácticas y semánticas

Este lenguaje, basado en el álgebra relacional y el cálculo relacional, actúa de interfaz entre el usuario y la base de datos y facilita realizar todas las operaciones permitidas. El lenguaje fue diseñado para que, mediante un número muy reducido de comandos y una sintaxis simple, fuese capaz de realizar un gran número de operaciones.

La curva de aprendizaje de SQL es realmente rápida. Además, SQL es bastante flexible, en el sentido de que cláusulas SQL pueden ser anidadas indefinidamente dentro de otras cláusulas SQL, facilitando así las consultas que utilizan varias relaciones, vistas u otras consultas.

Además de poder ser usado directamente, es decir, en modo comando, desde el SGBD, SQL puede ser usado desde otros lenguajes de programación de tercera generación, tales como C, para poder acceder a los datos de la base de datos y usarlos para cualquier fin en el programa. Cuando SQL es usado de este modo se le denomina SQL embebido. Esta característica amplía enormemente las posibilidades del modelo relacional.

Su aprendizaje no solo sirve para esta aplicación sino, también, para todas las existentes en el mercado que soporten este lenguaje ya que es un lenguaje estándar por haberse visto consolidado por el Instituto Americano de Normas[7] (ANSI) y por la Organización de Estándares Internacional[8] (ISO).

LOS ORÍGENES DE SQL

La historia de SQL empieza en 1974 con la definición, por parte de Donald Chamberlin y de otras personas que trabajaban en los laboratorios de investigación de IBM[9], de un lenguaje para la especificación de las características de las bases de datos que adoptaban el modelo relacional. Este lenguaje se llamaba SEQUEL (Structured English Query Language) y cuyas experimentaciones lo convirtieron finalmente en SQL y en un estándar para las bases de datos relacionales.

[7] http://www.ansi.org/, Último acceso 19-05-2011
[8] http://www.iso.org/, Último acceso 19-05-2011
[9] http://www.ibm.com/, Último acceso 19-05-2011

SQL se convierte así en el lenguaje de manipulación de datos estándar de ANSI, poseedor de un componente de definición de datos, más potente.

LENGUAJE DE MANIPULACIÓN DE DATOS (LMD)

El Lenguaje de Manipulación de Datos *DML* (*del inglés Data Manipulation/ Management Language*) es el encargado de facilitar a los usuarios el acceso y manipulación de los datos mediante comandos básicos:

Operaciones de Tablas: *CREATE, DROP, TRUNCATE*.

Operaciones de Tuplas*: INSERT, UPDATE DELETE*.

Disparadores (*del inglés triggers*): se ejecutan tras la evaluación de una condición.

LENGUAJE DE DEFINICIÓN DE DATOS (LDD)

El Lenguaje de Definición de Datos DDL (*del inglés Data Definition Language*) provee de los medios necesarios para definir los datos con precisión, especificando una arquitectura de tres niveles: estructura lógica global, estructura interna, y definición de las estructuras externas.

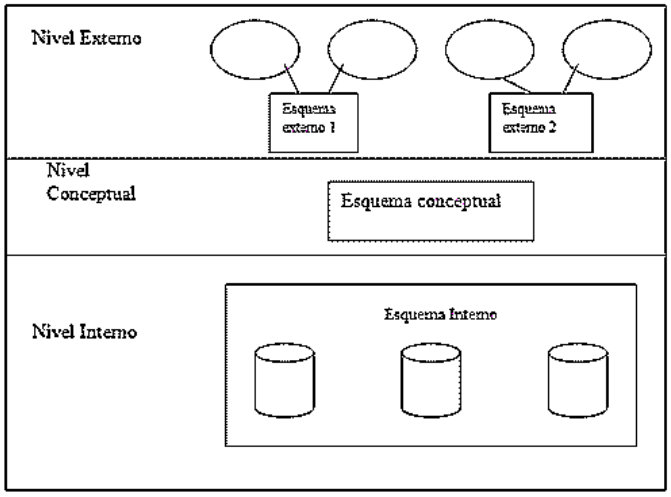


Figura 9. Arquitectura de 3 niveles.

2.2.6. CONCLUSIONES

Los sistemas de gestión de bases de datos suponen un pilar fundamental en cualquier tipo de PYME o de gran empresa ya que, por necesidad de eficiencia, estas requieren un control de información fiable y consistente que permita manipularla según las necesidades del negocio. Además, debido a que la cantidad de información crece año tras año de manera notable, imponen que estos sistemas tengan un carácter dinámico y permitan a los diseñadores y programadores readaptarlas a estos cambios y a las necesidades y requisitos del cliente.

Por otro lado, el espectro de opciones existente para decidir qué base de datos elegir es muy grande y pueden ser de carácter tanto comercial, gratuitas u Opensource. Esto lo que provoca es que un sistema empresarial como el actual, que se rige por la competencia y demanda de productos, aporte este toque de intencionalidad de mejora, de ser propietario del mejor producto en el mercado, tener afán de superación y de adaptación continua a la demanda de los usuarios.

En función de las necesidades de la empresa, del tamaño de información que se maneje en ella y de los conocimientos de su plantilla, puede elegirse el SGBD que mejor pueda suplir dichas necesidades. Se podría afirmar pues que no es la empresa la que se adapta sino que son las bases de datos las que cambian continuamente amoldándose a las necesidades del mercado.

2.3. INTERFACES RICAS

2.3.1. INTERFACES DE USUARIO

Los avances de la ciencia y la tecnología han puesto al hombre en un plano intermedio entre lo tangible e intangible computacionalmente hablando. Es ahora tan común el convivir con un ordenador diariamente que cada vez es más obligatorio optimizar y facilitar la interacción hombre-máquina a través de una adecuada interfaz (Interfaz de Usuario [25] o GUI), que le brinde tanto comodidad ,como eficiencia a la hora de cubrir sus necesidades.

CONCEPTO DE INTERFAZ DE USUARIO

Una interfaz de usuario, propiamente dicho, está constituido por aquellos objetos visuales que incluyen menús, ventanas, ratón, sonidos, creando canales de comunicación entre el usuario y la máquina, sirviendo como sistema de traducción entre los diferentes lenguajes de comunicación que emplean ambos involucrados y facilitando la interacción del usuario con los datos y el sistema de información.

La idea fundamental es que el usuario dirija el funcionamiento de la máquina mediante instrucciones de entrada asociadas a los objetos visuales sin preocuparse del código que exista por debajo de la interfaz ni de la arquitectura que lo sostiene. Además, actualmente, las interfaces de usuario constituyen unos de los principales elementos de un sistema operativo.

FACTORES A TENER EN CUENTA A LA HORA DE DISEÑAR UN INTERFAZ DE USUARIO

Al diseñar interfaces de usuario deben tenerse en cuenta las habilidades cognitivas y de percepción de las personas, y adaptarlas a ellas.

Así, una de las cosas más importantes que una interfaz puede hacer es reducir la dependencia de las personas de su propia memoria, no forzándoles a recordar cosas innecesariamente (por ejemplo, información que apareció en una pantalla anterior) o a

repetir operaciones ya realizadas (por ejemplo, introducir un mismo dato reiteradamente). Además, se pretende que el usuario aprenda a utilizar la aplicación lo más rápido posible. Algunas de las características objetivo del diseño de una interfaz de usuario son:

- **Velocidad de respuesta**: Tiempo necesario para realizar una operación.
- **Tasa de errores**: Hay que contemplar que los errores cometidos por el usuario para realizar y completar una operación sean mínimos.
- **Capacidad retentiva**: Cuánto recuerda el usuario de lo que ha hecho tras realizar un ciclo de operaciones a lo largo del tiempo.
- **Satisfacción**: Mide el grado de agrado del usuario con el sistema.
- **Adaptabilidad:** Cómo se adapta el usuario al cambio de una aplicación a la nueva.

CONCLUSIONES

Las Interfaces de Usuario, como vínculo de inmersión del hombre en el entorno de trabajo tecnológico actual, realzan su importancia en el desarrollo de nuevos productos, más eficaces, eficientes e interactivos, que es lo que el mercado demanda.

Además facilitan el uso de las aplicaciones, permitiendo al usuario aprender rápido y eficientemente. Esto convierte a las Interfaces de Usuario en un elemento a cuidar hoy en día y sobre el cual hay que tener especial cuidado a la hora de diseñar, ya que influirá fervientemente en la interacción del usuario con la máquina. Cuanto mejor sea y más intuitiva, mayor capacidad de trabajo tendrán los usuarios.

Es por todo esto que es evidente el papel fundamental a que juegan en el desarrollo de aplicaciones.

2.3.2. LA SOLUCIÓN DE MICROSOFT. EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

DEFINICIÓN DE XAML

XAML [26] (E**X**tensible **A**pplication **M**arkup **L**anguage) es la propuesta de Microsoft[10] a la necesidad de implementar interfaces gráficas de usuario dentro, o no, del entorno de desarrollo .NET.

Se trata es un lenguaje declarativo basado en la capacidad de extensibilidad del lenguaje XML y permite a su vez, a grandes rasgos, formar estructuras organizativas de objetos .NET en forma de árbol jerárquico, dónde unos objetos dependen directa o indirectamente del nodo padre al que pertenecen. La interpretación posterior con el motor de visualización WPF de Microsoft, y el marco de trabajo que ofrece, definirá la interfaz visual a la que el usuario tendrá acceso. A través de esta tendrá la posibilidad de interactuar, según el modelo de code behind, con los objetos previamente definidos mientras que toda la funcionalidad en C# residirá oculta al usuario y aportará el carácter dinámico en tiempo de ejecución de la aplicación.

Esta lógica será la que asocie la parte visual que aporta XAML y WPF con la parte funcional sin mezclar los ingredientes ya que XAML es independiente y puede, quizá en un futuro, ser la base estructural de otro motor que pueda interpretarlo y generar el resultado deseado de igual modo que hace WPF.

FRAMEWORK DE TRABAJO

Por todo lo dicho anteriormente, podemos afirmar y no es difícil ver que, el marco de desarrollo de aplicaciones .NET ofrece una dinámica de desarrollo estructural Modelo-Vista-Controlador donde se definen claramente las 3 capas de trabajo y desde las cuales se puede acceder sin ninguna restricción al resto de capas para conseguir un riguroso

[10] http://www.microsoft.com/ . Último acceso 20-05-2011

entrelazado y libertad de interacción entre ellas. XAML y WPF pertenecerían a la Vista, lo cual que evitaría que le usuario necesitara entender lo que ocurre realmente por debajo, por ejemplo, si pulsa un botón porque quiera ejecutar una sentencia SQL. Las capas del Controlador y del Modelo se encargarían de ejecutar de forma "invisible" para el usuario las funcionalidades necesarias para realizar la operación, relacionando de manera lógica los objetos que estuvieran implicados en dicha operación.

OBJETIVO DE XAML

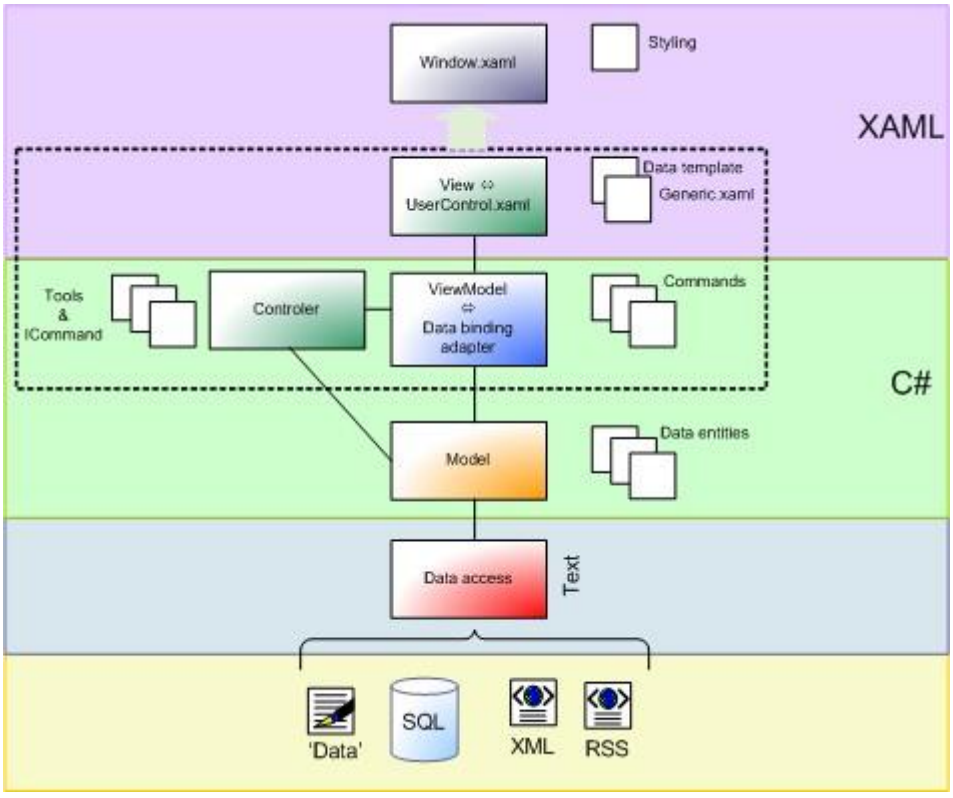


Figura 10. Arquitectura lógica MVC.

El objetivo principal y final de XAML, como lenguaje declarativo, es ofrecer una forma muy consistente y amigable de permitir que usuarios convencionales puedan emplear aplicaciones que, independientemente de la complejidad de la lógica de negocio que se utilice y de sus conocimientos, puedan desarrollar eficaz y productivamente sus obligaciones. A día de hoy, puesto que en el 80% de las empresas es necesario el manejo de una aplicación que gestione el negocio, esto supone una necesidad a cubrir que, sobradamente XAML proporciona.

EXTENSIÓN A INTERFACES DE APLICACIONES .NET

.NET se ha convertido en un referente como framework de trabajo en la mayoría de empresas, ya no sólo porque sea un producto de Microsoft, organización empresarial que en pocos años se ha reafirmado como uno de los pilares fundamentales en el entorno de desarrollo de sistemas de información, sino porque .NET en sí tiene y ofrece todo lo necesario para desarrollar rápidamente aplicaciones de cualquier tipo y complejidad. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones, permitiendo una mejor integración entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

¿En qué sentido se puede confirmar la estrecha relación entre XAML, WPF y .NET?

Ya se ha visto la estructura MVC que conforman XAML WPF y SQL y que Visual Studio se encarga de engranar como piezas de un reloj permitiendo una asociación libre entre ellas.

Todo lo que se implementa en XAML puede ser expresado y traducido utilizando C#, o cualquier lenguaje soportado por Visual Studio, lo cual aporta una reducción necesaria de complejidad para procesarlo ya que, recordamos, es una extensión del lenguaje XML. Esto supone un beneficio enorme a los desarrolladores, permitiéndoles compartir, editar y actualizar los contenidos con libertad sin necesidad de compilar. Y el único responsable de que toda esta increíble maquinaria funcione es Microsoft.

CONCLUSIONES

La integración de XAML y WPF como lenguaje de etiquetado y declarativo dentro del marco de desarrollo .NET proporciona legibilidad y capacidad de modificación según las necesidades del usuario convirtiendo la interfaz resultante en una interfaz rica y maleable. Y decimos maleable desde el punto de vista de que no es un lenguaje estricto y riguroso sino que permite la creación de objetos personalizados, permitiendo extender, mejorar y personalizar la funcionalidad de dichos controles XAML. Esto permitirá a los desarrolladores definir entornos gráficos con una apariencia particular y distinta en muchos casos; se podría

decir que XAML aporta los cimientos y material de construcción base y el usuario decide cuál va a ser el diseño final.

El carácter extensible es una propiedad muy importante de XAML ya que la funcionalidad de muchos de los controles visuales dependerá en gran medida de la lógica de la aplicación y de qué se quiere conseguir cuando se interactúa con ellos. Por ejemplo, un botón puede enviar un formulario, validar un campo, colapsar un panel o realizar cualquier tipo de acción que se desee implementar con cierto lenguaje soportado por la plataforma .NET.

En relación con WPF, XAML debido a su versatilidad permite que muchos controles que no estén definidos en WPF o que su motor no reconozca, puedan ser “disfrazados” e implementados de manera que los interprete y pueda compilar, ampliando desde el punto de vista de XAML la capacidad de carga visual de WPF a controles que realmente no puede traducir.

2.4. INTEGRACIÓN DE APLICACIONES DE CLIENTE COMO COMPONENTE

2.4.1. DESCRIPCIÓN

La creciente necesidad de realizar sistemas complejos en cortos periodos de tiempo, a la vez que con menores esfuerzos tanto humanos como económicos, está favoreciendo el avance de lo que se conoce como Desarrollo de Software Basado en Componentes (DSBC). Esta nueva disciplina se apoya en componentes software ya desarrollado, que son combinados adecuadamente para satisfacer los requisitos del sistema. Es por tanto una cuestión de **reutilización**.

Construir una aplicación se convierte por tanto en la búsqueda y ensamblaje de piezas prefabricadas, desarrolladas en su mayoría por terceras casas, y cuyo código no puede modificarse. Bajo este nuevo planteamiento, cobran especial interés los procesos de búsqueda y selección de los componentes apropiados para construir las aplicaciones. En este sentido, la tecnología de componentes ofrece ya soluciones robustas para muchos de los problemas que se plantean en la construcción de grandes sistemas de software y, de hecho, vivimos inmersos en una creciente integración como componentes del software.

Los principales esfuerzos de la comunidad de software en estos temas se han basado hasta ahora en los aspectos funcionales de los componentes, es decir, en la funcionalidad que ofrecen. Se enfrentan a continuos retos para conseguir que una amplia variedad de software de muchos proveedores funcione conjuntamente y este aspecto es crucial para tener éxito en la simplificación de los procesos de negocios, para estar más cerca de los clientes y negocios, o para hacer fusiones y adquisiciones exitosamente.

Cuando se conecta con sistemas de socios, cuando se acceden datos de un mainframe, cuando se conecte a aplicaciones escritas en diferentes lenguajes de programación o trate de

iniciar una sesión en diferentes sistemas hay que conseguir que funcionen tecnologías heterogéneas al tiempo que se reducen costes.

La solución que ha probado efectividad constantemente, y la que rinde el mayor éxito para los desarrolladores hoy en día, es la del compromiso de la interoperabilidad. Esto significa el permitir que diferentes tipos de aplicaciones y sistemas hagan lo que mejor saben hacer, mientras coinciden en un "contrato" común de cómo los diferentes sistemas se pueden comunicar para intercambiar datos. Sin lugar a dudas, un paquete consistente que ofrece este nivel de heterogeneidad y de interoperabilidad es Microsoft Office.

Una de las innovaciones de Microsoft Office es la flexibilidad para integrar componentes desarrollados por compañías terceras. Se emplean técnicas como desarrollo de servicios de análisis y manejo de datos, como es nuestro caso. Esto permite que, sin dejar de utilizar el paquete de ofimática ofrecido por Microsoft, las empresas puedan ampliar su funcionalidad y facilitar la integración y combinación de su software, potenciando su efectividad, eficacia y calidad.

2.4.2. COMPONENTES DE SOFTWARE

En el contexto de Ingeniería de Software, un "activo" reutilizable es un producto diseñado expresamente para ser empleado de forma recurrente en el desarrollo de muchos sistemas y aplicaciones. Ejemplos de activos reutilizables son: algoritmos, patrones de diseño, esquemas de base de datos, arquitecturas de software, especificaciones de requerimientos, de diseño y de prueba, entre otros. En los últimos años, como resultado de presiones crecientes sobre la industria del software orientadas a reducir drásticamente el costo y tiempo de desarrollo de sistemas y aplicaciones, sin afectar los niveles de calidad del producto, ha surgido un nuevo activo reutilizable denominado Componente de Software.

Recientemente, el Instituto de Ingeniería de Software[11] (SEI, Software Engineering Institute) formuló una definición con el propósito de consolidar las diferentes opiniones acerca de lo que debía ser un componente de software. Según el SEI, un componente es "una implementación

[11] http://sei.cmu.edu/. Último acceso 22-05-2011

opaca de funcionalidad, sujeta a composición por terceros y que cumple con un modelo de componentes".

Con respecto al primer aspecto, un componente se considera una implementación opaca debido a que su distribución predominantemente es en formato binario y sus consumidores lo utilizan como una "caja negra" a través de su interfaz. Dicho aspecto está alineado con el principio de encapsulamiento de la programación orientada a objetos.

Por otra parte, la composición por terceros implica que los componentes son intrínsecamente reutilizables debido a que un sistema basado en componentes puede ser ensamblado con relativa facilidad por un integrador empleando componentes suministrados por múltiples proveedores independientes.

Finalmente, la coordinación e interacción entre componentes exige un marco regulatorio estandarizado (modelo de componentes) que establece la infraestructura de software requerida (framework) y las convenciones y restricciones de diseño de los mismos.

Tal como lo refleja la definición anterior, un componente de software puede ser visto desde dos perspectivas distintas, como:

Implementación: los componentes se pueden ensamblar y desplegar para crear sistemas y aplicaciones que se ejecutan en un computador.

Abstracción de arquitectura: los componentes expresan las reglas de diseño que impone el modelo de componentes.

Una de las características más importantes de los componentes es que son reutilizables. Para ello los componentes deben satisfacer como mínimo el siguiente conjunto de características:

- **Identificable**: Un componente debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.

- **Accesible sólo a través de su interfaz**: El componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.
- **Servicios son invariantes:** Las operaciones que ofrece un componente, a través de su interfaz, no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz.
- **Documentado**: Un componente debe tener una documentación adecuada que facilite su búsqueda en repositorios de componentes, evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte.

Adicionalmente, para favorecer su reutilización es deseable que un componente sea:

- **Genérico**: Sus servicios pueden ser usados en una gran variedad de aplicaciones.
- **Auto contenido**: Es conveniente que un componente dependa lo menos posible de otros componentes para cumplir su función de forma tal que pueda ser desarrollado, probado, optimizado, utilizado, entendido y modificado individualmente.
- **Mantenido**: Es deseable que un componente, como toda pieza de software, esté inmerso en un proceso de mejoramiento continuo que le garantice al integrador nuevas versiones que incluyan correctivos, optimizaciones y nuevas características. Esto contribuye a que dicho componente sea seleccionado con mayor frecuencia para formar parte de sistemas de software.
- **Independiente:** De la plataforma, hardware y sistema operativo, del lenguaje de programación y de las herramientas de desarrollo. Existen diversas plataformas de cómputo de uso frecuente, como Windows/Intel, Solaris/Sparc, OSX/PPC, Linux/Intel, y es deseable que un componente pueda ejecutarse en todas ellas. Asimismo, ya que existe una amplia gama de lenguajes de programación y herramientas de desarrollo, es natural que encontremos componentes escritos empleando lenguajes y herramientas de la preferencia del programador, por lo tanto es deseable que dichas preferencias no limiten el uso de los componentes.
- **Reutilización dinámica**: Puede ser cargado en tiempo de ejecución en una aplicación.

- **Certificado**: El componente puede ser certificado por una agencia de software independiente o mediante la aplicación de modelos de auto-certificación que le permiten al comprador del componente determinar la calidad del software adquirido.

2.4.3. MECANISMOS DE COMPOSICIÓN DE SOFTWARE

Bajo el modelo de desarrollo de software basado en componentes, las nuevas aplicaciones se construyen mediante la integración o composición de componentes.

"La composición de software es el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces (de composición)."

Nótese que se hace especial énfasis en las interfaces como elementos fundamentales para lograr la composición de componentes. Además de los componentes, los framework también se consideran entidades sujetas a composición. En consecuencia, existen tres clases principales de interacción en los sistemas basados en componentes:

- **Componente-Componente (C-C):** permite la interacción entre componentes.
- **Componente-Framework (C-F):** posibilita las interacciones entre el framework y sus componentes. Dicha interacción permite que el framework administre los recursos de los componentes.
- **Framework-Framework (F-F):** posibilita las interacciones entre frameworks y permiten la composición de componentes desplegados en frameworks heterogéneos. Estos contratos puede ser clasificados como contratos de interoperabilidad.

La forma de materializar la composición entre componentes depende de los mecanismos especificados por su modelo de programación. Típicamente, los modelos de componentes se basan en tecnologías orientadas a objetos, por lo tanto los mecanismos de composición emplean algunas características tales como relaciones entre clases (especialización, agregación, asociación y uso), polimorfismo y enlace dinámico. Adicionalmente, dichos mecanismo de composición típicamente se describen mediante el uso de patrones de diseño.

Las tecnologías de componentes no distribuidos, típicamente asociados con aplicaciones de escritorio, por ejemplo Controles ActiveX y JavaBeans, hacen uso extensivo de características orientadas a objetos dentro de sus mecanismos de composición. Por el contrario, en la composición de componentes distribuidos, por ejemplo Enterprise JavaBeans y .NET, principalmente se emplean relaciones de uso, asociación y agregación.

2.4.4. ANALÍTICA

Entre las razones para integrar este proyecto en la suite de productividad de Microsoft Office se pueden destacar varios factores que, en conjunto, extienden la funcionalidad del software de empresas para sus propios propósitos. Como consecuencia de esto, su software tendrá mayor calidad, será más confiable, motivará a los programadores a afrontar nuevos desarrollos y a encontrar soluciones. La razón más importante, ya que es la que es el principal objetivo de este proyecto, es que también permite conjugar lenguajes de programación y entornos de desarrollo.

Además, la integración de un motor de datos en lenguaje SQL y desarrollado en el entorno .NET como complemento Office supone una gran ventaja, tanto en costes como en adaptabilidad:

Costes: Reducción significativa del costo de propiedad. Microsoft Office provee a los administradores de un control central y de una gran flexibilidad reducir el costo del soporte al usuario final.

Experiencia: Simplifica en gran medida la experiencia del usuario con el software.

La mejora de eficiencia en el ambiente de trabajo, la cual permitiría a los clientes una rápida y mayor adaptabilidad a las demandas de mercado y del usuario final y que exigen cambios en el proyecto, reutilización de componentes, etc., es otra de las características que la integración de componentes ofrece a los sistemas de negocio.

Los clientes demandan retroalimentación desde un punto de vista funcional. Esto significa que a mayor integración, mayor capacidad de uso y aumento en la competitividad de cara al

mercado. Para enfrentarse a la competitividad es de vital importancia encontrar un equilibrio entre productividad, control de costes, facilidad de despliegue y migración de software; y la complejidad de encontrar este equilibrio es ofrecido por Microsoft Office.

2.4.5. LA SOLUCIÓN COMO COMPONENTE COM OFFICE. VISUAL STUDIO TOOLS FOR OFFICE

El problema que se plantea es cómo conseguir esta heterogeneidad y esta integración. Para dar solución a este problema, existe una extensión de MS Office que permite crear soluciones basadas en Office desde un entorno de desarrollo .NET. Este extensor es VSTO [27] (**V**isual **S**tudio **T**ools for **O**ffice). Se trata de un conjunto eficaz de herramientas y características integrado dentro de Visual Studio que permite a los desarrolladores ampliar y personalizar las aplicaciones de Microsoft Office mediante el uso de Visual Basic, C#, WPF...

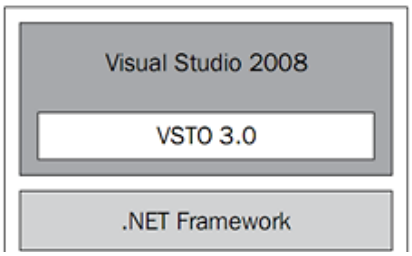


Figura 11. Estructura integrada VSTO.

"VSTO es el puente que nos posibilita integrar Office en nuestro sistema de negocio."

Admite el uso de controles de Windows Forms, incluye diseñadores de flujo de trabajo visual y dispone de diseñadores visuales para Word 2007 y Excel 2007 que aparecen dentro del IDE de Visual Studio. Con todo esto se consigue que el documento actúe como una superficie de diseño visual.

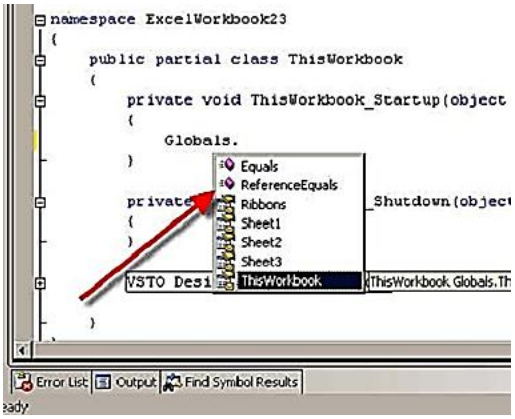


Figura 12. Integración Ribbon en Visual Studio.

¿Por qué se ha elegido Visual Studio Tools para Office?

Hay muchas razones relevantes para decidirse por VSTO y que dependen directamente con propiedades que ofrece Visual Studio. Algunas de estas características son:

- **Mejora de la productividad:** Microsoft Visual Studio es uno de los entornos de desarrollo más sofisticado y rico del mercado ya que mejora el rendimiento de codificación y testeo (posee el debugger es el más potente del mundo).
- **Seguridad:** Posee un modelo de seguridad construido sobre el CLR ya que se ejecuta en un dominio fiable y no puede ser accedido desde el sistema de archives del usuario.
- **Facilidad de versionado:** Uno de los beneficios principales del CLR es que el usuario puede crear muchas versiones de la misma aplicación y ejecutarlas en cliente sin preocuparse por la aparición de conflictos.
- **Reutilización de código:** VSTO, debido a su estructura modular, facilita la reutilización de bloques de código en otras aplicaciones.
- **Integración:** El marco de desarrollo .NET facilita la integración con otras aplicaciones.

En el caso de este proyecto, se utilizará VSTO para incluir un control en el interfaz de Office que lanzará la aplicación que posteriormente se comunicará con una base de datos a través de otra interfaz con su correspondiente funcionalidad.

2.5. CONCLUSIONES AL ESTADO DEL ARTE

Día tras día, la demanda de integración de aplicaciones en entornos de trabajo habituales, tanto en empresas como a nivel usuario se ve incrementada. Los usuarios están cansados de que, para realizar tres tareas, necesite tener instaladas en su máquina tres aplicaciones diferentes y que, probablemente, no le aseguren la posibilidad de acoplar las salidas de uno y de otro, teniendo que readaptar continuamente lo que necesita y obtiene de cada una de ellas, por ejemplo, obtener un .doc en Office y tener que pasarle un conversor para pasarlo a formato .pdf. Este proceso supone una pérdida de tiempo innecesaria que una buena integración de componentes entre aplicaciones podría evitar.

Si nos detenemos a pensar, cada vez se desarrollan y distribuyen más componentes como plugins, extensiones, complementos de aplicación, etc, que dan cobertura a la necesidad de integración de componentes en aplicaciones y dan una solución al problema. Por tanto, es obviamente necesario concienciarse y unificar esfuerzos para facilitar la integración de aplicaciones y de sus funcionalidades.

CAPÍTULO III

ANÁLISIS DEL PROBLEMA

3.1. PROBLEMÁTICA AFRONTADA

Hoy en día son muchas empresas, por no decir todas, las que disponen de un sistema gestor de bases de datos con el que acceder a la información que almacenan, información de suma importancia tales como datos de clientes, cuentas, etc. Pero no se han planteado la posibilidad de que cualquier empleado, no solo un ingeniero de software capacitado, sea el que se encargue de realizar las operaciones sobre dicha información y que los informáticos puedan dedicarse a tareas más cercanas a la ingeniería.

El problema es cómo conseguir desarrollar una aplicación que consiga este propósito. Se plantean aquí varias preguntas, entre ellas cómo hacerlo, en qué entorno de desarrollo, con qué herramientas y si la alternativa seleccionada será las más apropiada o habrá que revertir acciones ya tomadas. Además se pretende obtener un producto que pueda acoplarse a aplicaciones ya existentes así como a distintas plataformas. En el siguiente capítulo se pretende dar respuesta a estas preguntas.

Para abordar este proyecto, existe una clara limitación de personal de desarrollo, ya que es una única persona la que se encargará del desarrollo de todas y cada una de las fases del mismo. Además será él solo quien desempeñe las funciones de Jefe de Proyecto, Analista Funcional, Programador y Administrados de la base de datos. Esto generará un retraso importante ya que se trata de una persona que no posee experiencia en todos los roles involucrados y deberá adaptarse continuamente a ellos, evitando entorpecer y retrasar el ciclo de desarrollo del proyecto.

Otro problema crítico, identificado en esta fase previa al comienzo del proyecto, es la inexperiencia del programador para desarrollar aplicaciones pese a su experiencia laboral junto con la de desempeñar funciones dentro del equipo de desarrollo que nunca ha realizado, como la de Jefe de Proyecto. Sería necesario un periodo de formación con respecto a tecnologías nuevas aplicadas a este proyecto y de desempeño, lo que supondría una inversión de capital y tiempo del que no se dispone.

Además, existe un problema en el componente de integración de un entorno de desarrollo multiplataforma. No todas las empresas utilizan el mismo sistema operativo, pero sí emplean un sistema gestor de bases de datos que puede ser multiplataforma como SQL. Entonces, ¿no habría que desarrollar la aplicación en un entorno de desarrollo completamente multiplataforma? No todos los entornos lo son, por ejemplo .NET es exclusivo de Microsoft y las aplicaciones desarrolladas a través de dicho entorno solo ejecutan en sistemas Win3, por el momento.

3.2. OBJETIVOS DEL PROYECTO

El objetivo de esta investigación será en primer lugar conseguir integrar la aplicación como un componente integrado en el paquete de productividad Microsoft Office.

El siguiente objetivo será reducir la brecha existente entre tareas técnicas computacionales y ordenes simples en el lenguaje cotidiano.

La necesidad de un sistema de este tipo, es grande, pues la gran cantidad de usuarios de servicios de computación desconoce la estructura sintáctica de diferentes herramientas que por el momento son conocidas por grupos comúnmente llamados de élite.

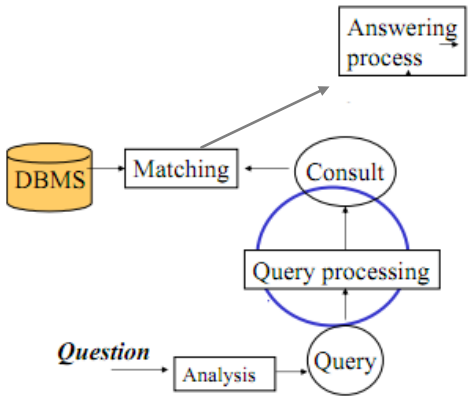


Figura 13. Arquitectura de los INLBD.

Como resultado se obtendrá un componente integrado que será capaz de conseguir convertir una consulta en lenguaje natural a SQL satisfactoriamente y que pueda ser reutilizada sin importar el dominio, es decir, que el administrador de la ILNBD o el usuario no tengan que hacer modificaciones al código o realizar tediosas configuraciones para que ésta funcione en un dominio diferente.

3.3. PROPUESTAS DE MEJORA A LOS PROYECTOS EXISTENTES

A continuación se van a exponer una serie de posibles mejoras aplicables a los sistemas descritos en el apartado anterior.

Los ILNBD mencionados anteriormente, por su contexto temporal, tienen el problema de que no han aprovechado, o tenido la posibilidad de hacerlo, las herramientas principales que hoy en día encontramos en el mercado y que facilitan enormemente los nuevos desarrollos, como por ejemplo entornos de desarrollo sofisticados, robustos y claramente definidos *(ver 4.3. TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO DE NQPL y 4.4. HERRAMIENTAS EMPLEADAS EN EL DESARROLLO").* Además, se han implementado como aplicaciones independientes o standalones, sin capacidad de integración y que únicamente se van a poder ejecutar sobre una plataforma de desarrollo, incumpliendo con una de las características de software de calidad como es la portabilidad.

En el desarrollo que vamos a realizar, se trata de un componente Win32 pero que, al estar desarrollado sobre .NET, y gracias al entorno de desarrollo multiplataforma Mono, podríamos ejecutar la aplicación sobre otros sistemas como por ejemplo GNU/Linux, FreeBSD, UNIX, Mac OS X, Solaris y obviamente Windows.

Además, el empleo de WPF y XAML en este proyecto, mejora en gran medida la interfaz visual y con ello la experiencia del usuario. En los sistemas existentes, debido a que están orientadas a usuarios expertos, las interfaces pierden importancia ya que, un usuario de esas características, está acostumbrado a desenvolverse en pantallas complicadas, llenas de campos, sin importarles la apariencia.

El componente multilenguaje de NQPL rompe con la brecha marcada por la mayoría de los traductores a lenguaje SQL ya que emplean módulos de descomposición semántica y generación de sentencias intermedias que impide la traducción de lenguajes cuyas estructuras gramaticales

sean diferentes (posición en la frase del verbo, del sujeto, de los adjetivos, tiempos verbales, etc.). En este proyecto se va a generar un lenguaje intermedio en función del lenguaje de entrada, es decir, los algoritmos de traducción serán diferentes en función de la cultura del usuario del sistema, generándose el mismo resultado SQL.

Otra ventaja que presenta con respecto a los sistemas existentes es que Visual Studio, por la cantidad de drivers de conexión de que dispone, permitiría, en el módulo de traducción, utilizar la misma expresión de entrada para traducirla a un lenguaje y, transportar la sentencia de dicho lenguaje a otro, para generar 2 salidas en lenguajes diferentes con un único análisis gramatical. Por ejemplo, se podría traducir una expresión a SQL y de SQL a una sentencia que recupere datos de una tabla Access.

3.4. SOLUCIÓN PROPUESTA

El desarrollo de un complemento Office que aporte al usuario la funcionalidad de un sistema basado en SQL y que reúna las ventajas que aporta XAML y WPF, tanto visuales para el usuario final como de implementación para los desarrolladores y todo mediante uno de los entornos más potentes del mercado como Visual Studio .NET, supone una gran acometida y reto que, funcionalmente, puede dar solución a muchos problemas de integridad, seguridad, usabilidad y sobre todo de portabilidad y adaptabilidad en cualquier máquina de hoy en día.

Así pues, un usuario que necesite interactuar con una base de datos desde un entorno de ejecución Office conocido y amigable, no tendría la necesidad de ejecutar más aplicaciones complejas y que requirieran un proceso de aprendizaje y adaptabilidad, pudiendo realizar todo su trabajo desde la misma ventana de aplicación, con un consumo de recursos de la máquina y software óptimos.

Además de esto, poder interactuar con una base de datos tal cual lo haría con su entorno habitual y empleando un lenguaje que le es conocido, sin conocer sintaxis de lenguajes de programación, y mediante el cual va a ser capaz de realizar cualquier tipo de operación permitida por dicho lenguaje, supone una solución para aquellos usuarios de la aplicación que no tengan unos conocimientos sólidos de lenguajes de programación. Esto conseguirá que la aplicación no limite su alcance a un entorno empresarial, sino que cualquier usuario que esté en predisposición de mantener una base de datos casera para uso particular, mediante esta aplicación, pueda evitar tener que aprender las bases de SQL y del manejo de instrucciones de manipulación de datos para poder operar con agilidad y facilidad con dichos datos, suponiéndole un ahorro de tiempo y costes de aprendizaje que quizá no pueda permitirse.

Por otro lado, la aplicación tendrá cierto carácter didáctico, es decir, a la vez que el usuario ejecute instrucciones basadas en el lenguaje natural sobre la base de datos podrá simultáneamente ver la traducción de dicha sentencia a lenguaje SQL, con lo que llegará un momento en que, debido al uso, empiece a comprender cómo funciona.

CAPÍTULO IV

DESARROLLO DE LA SOLUCIÓN

4.1. INTRODUCCIÓN

Antes de empezar hablar acerca de que podría consistir la calidad de un producto y/o servicio, es importante definir qué es lo que se entiende por calidad, a qué es aplicable y de qué forma puede ser relacionada con productos software.

La calidad se puede definir como "una característica o atributo de una cosa". De esta forma se podría decir que la calidad de los productos puede medirse como una comparación de sus características y atributos.

4.1.1. CICLO DE VIDA DE SOFTWARE

El desarrollo de productos software no está ausente de ofrecer calidad, es por esto que día a día las organizaciones y especialmente las productoras de software se preocupan por implementar mecanismos y estrategias que les conlleven a garantizar:

- La producción de código más fiable.
- Mayor nivel de calidad en todo el ciclo de la producción del software.
- Satisfacción de sus clientes y mejora de su ventaja competitiva.

Uno de los problemas más importantes en cualquier departamento de sistemas es definir un marco de referencia común que pueda ser empleado por todo el equipo de desarrollo y que defina los procesos, actividades y tareas a desarrollar. En este caso se ha optado por Métrica 3 como marco de desarrollo de software.

La norma IEEE 1074 define el ciclo de vida del software como una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento de software.

La norma ISO2 12-207-1 lo define como un marco de referencia que contiene procesos, actividades y tareas involucradas en el desarrollo, explotación y mantenimiento de un producto software, desde la definición de los requisitos hasta la finalización de su uso.

El ciclo de vida abarca por tanto toda la vida del sistema, desde su concepción hasta que se deje de utilizar. Existen modelos aplicados al ciclo de vida de software, entre los que podemos mencionar como principales el Modelo en cascada, Modelo incremental, Modelo en espiral y el Modelo orientado a objetos. Cada uno de estos modelos plantea el desarrollo de las tareas definidas anteriormente de un modo distinto a los otros modelos, con sus ventajas y desventajas.

El por qué se ha optado por aplicar el ciclo de vida en espiral se debe principalmente porque durante todo el ciclo de vida del software se pueden ir aplicando actividades:

- No necesita una definición completa de los requisitos para empezar a funcionar.
- Al entregar productos desde el final de la primera iteración es más fácil validar los requisitos.
- El riesgo en general es menor, porque si todo se hace mal, solo se ha perdido el tiempo y recursos invertidos en una iteración (las anteriores iteraciones están bien).
- El riesgo de sufrir retrasos es menor, ya que al identificar los problemas en etapas tempranas hay tiempo de subsanarlos.

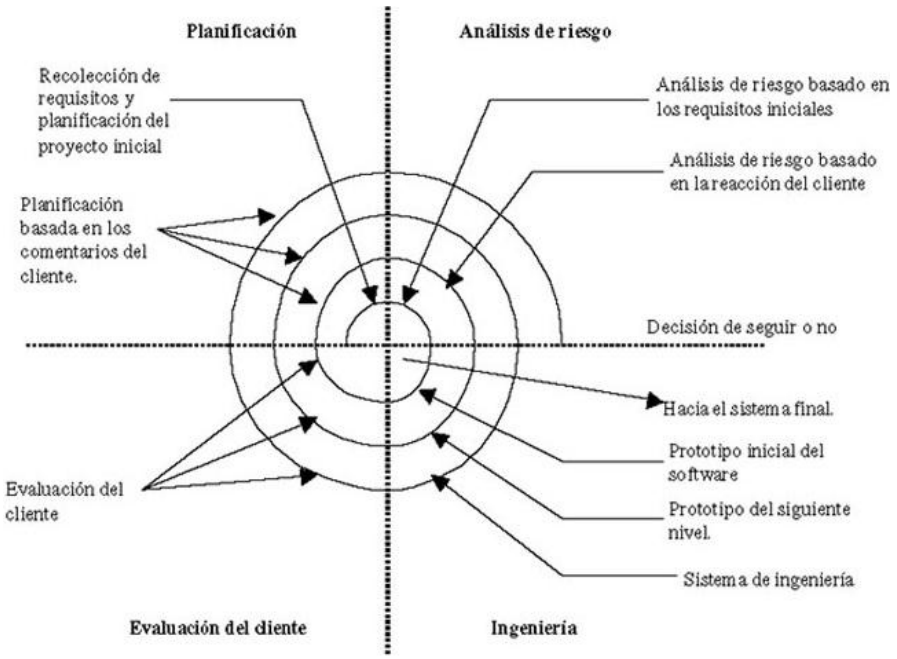


Figura 14. Ciclo de vida en espiral.

4.2. METODOLOGÍA

4.2.1. DEFINICIÓN

La dirección del presente proyecto, propuso seguir la metodología conocida como Métrica versión 3 ya que nos ofrece un marco sistemático sobre una serie de actividades a seguir para el soporte al ciclo de vida del software.

Métrica, contempla el desarrollo de sistemas de información que se adapten y utilicen las nuevas tecnologías siguiendo un plan de gestión que garantiza el cumplimiento de objetivos en términos de calidad, costos y plazos de tiempo. Ésta metodología, tiene objetivos orientados a la mejora de los procesos dentro de una empresa, tales como la implantación de Sistemas de Información que satisfagan las necesidades de los usuarios dotando de una importancia mayor al análisis de requisitos, mejorar la productividad del departamento de Sistemas, mejorando la adaptabilidad a cambios contemplando como base, la reutilización del trabajo en la medida de lo posible.

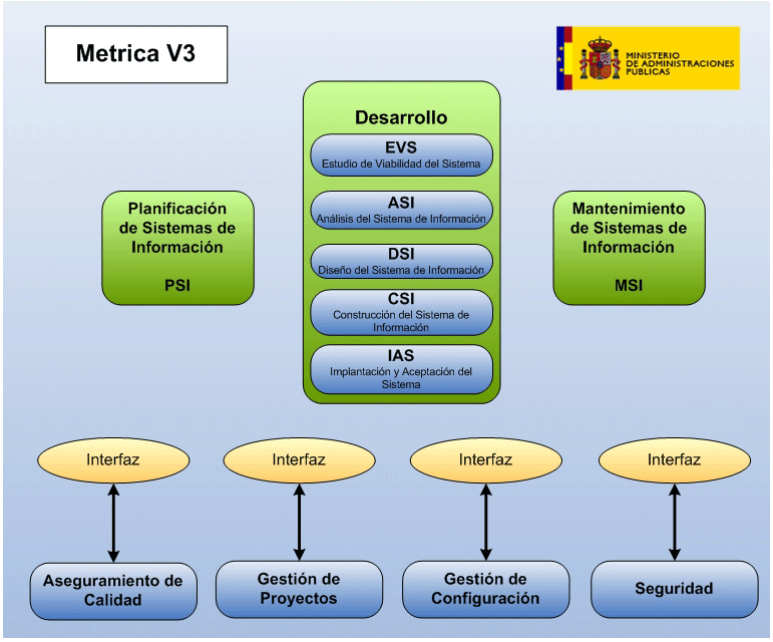


Figura 15. Esquema Métrica 3.

Si bien Métrica está orientada a la creación de Sistemas de Información, ofrece un vasto soporte a la creación de software en general, sustentado en una documentación muy completa

dividida en procesos (en alto nivel) y tareas (en el nivel más bajo) de las cuales se detalla el contenido, dejando en claro qué acciones, técnicas y prácticas han de llevarse a cabo por los participantes para completar una lista de productos. Al final de cada etapa, Métrica contempla una revisión de las actividades realizadas a fin de garantizar el cumplimiento de los objetivos y mantener al cliente informado sobre los avances; del mismo modo verificar la solución a las necesidades planteadas en un inicio y así, poder avanzar a las etapas subsecuentes.

¿Por qué Métrica 3?

Las razones por las cuales, se eligió esta metodología tienen fundamento en la especificación sucesiva de los distintos productos obtenidos de cada actividad, en su documentación ordenada y bien cimentada además de que permite mantener un seguimiento del trabajo realizado en cada fase.

4.2.2. OBJETIVOS DE MÉTRICA 3

Los principales objetivos que se conseguirán alcanzar a través del marco de desarrollo de Métrica 3 son los siguientes:

- Definir un Sistema de Información Organizativo a través de la especificación de un marco de desarrollo estratégico.
- Aportar un amplio soporte de las necesidades del usuario en la labor de análisis de requisitos del software a desarrollar consiguiendo un producto más afín y cercano a lo que se desea elaborar.
- Amplificar la capacidad de entendimiento y comunicación entre los participantes en la fase de producción a lo largo de todo el proceso de ciclo de vida del proyecto siempre teniendo muy en cuenta el papel que desempeña cada uno de los componentes del equipo de desarrollo, la fase de desarrollo que se esté elaborando y la distribución de responsabilidades.
- Facilitar el mantenimiento y reúso del producto desarrollado durante el periodo de mantenimiento y comercialización.

¿Qué son la serie de estándares ISO 9000?

La Organización Internacional para la Estandarización, mejor conocida como ISO, es la agencia especializada en estandarización, conformada por representantes de los cuerpos normalizadores, fue establecida oficialmente el 23 de febrero de 1947 con el objeto de promover la estandarización internacional, de tal manera que se facilitara el intercambio internacional de bienes y servicios casi como el desarrollo científico y tecnológico. Actualmente abarca los estándares nacionales de 91 países. En los Estados Unidos, la representación se llama *The **A**merican **N**ational **S**tandards **I**nstitute (ANSI).*

Las series de ISO 9000 son un grupo de 5 individuales, pero relacionadas, estándares internacionales de administración de la calidad y aseguramiento de calidad. Estos estándares fueron desarrollados para documentar efectivamente los elementos de sistemas de calidad que son instrumentados para mantener un sistema eficiente de calidad en la empresa sin que especifiquen la tecnología que debe ser aplicada para la instrumentación de los elementos del sistema de calidad.

Una estrategia mundialmente adoptada, ha sido la implementación del modelo de evaluación y mejora del proceso de software diseñado por ISO 9000, específicamente la guía ISO 9001 ya que es la más comprensible, abarca el diseño, la implementación, y la instalación de sistemas y mejor se ajusta a este proyecto. A partir de la ISO 9001 se conseguirá obtener un modelo de calidad de software que nos diga qué hacer, no cómo, ya que el cómo estará influenciado por la metodología que se emplee y el objetivo del negocio.

A partir este modelo, se elaborará la estructura de MÉTRICA Versión 3, la cual distingue procesos principales, Planificación, Desarrollo y Mantenimiento, e interfaces, Gestión de Proyectos, Aseguramiento de la Calidad, Seguridad y Gestión de Proyectos, y cuyo objetivo es dar soporte al proyecto en los aspectos organizativos.

PROCESOS PRINCIPALES DE MÉTRICA VERSIÓN 3

MÉTRICA Versión 3 es un modelo orientado al proceso, ya que la tendencia en los estándares se encamina en este sentido y por ello se centra en la clasificación y definición procesos del ciclo de vida del software. Como punto de partida y atendiendo a dicha norma, MÉTRICA Versión 3 cubre el Proceso de Desarrollo y el Proceso de Mantenimiento de Sistemas de Información. y abarcar el desarrollo completo de Sistemas de Información sea cual sea su complejidad y magnitud. Por ello su estructura responde a desarrollos máximos y deberá adaptarse y dimensionarse en cada momento de acuerdo a las características particulares de cada proyecto.

La metodología descompone cada uno de los procesos en actividades, y estas a su vez en tareas. Para cada tarea se describe su contenido haciendo referencia a sus principales acciones, productos, técnicas, prácticas y participantes.

El orden asignado a las actividades no debe interpretarse como secuencia de realización, ya que éstas pueden realizarse en orden diferente a su numeración o bien en paralelo, como se muestra en los gráficos de cada proceso; es flexible y adaptable a cada proyecto software. Sin embargo, no se dará por completado un proceso hasta no haber finalizado todas las actividades del mismo y que son estipuladas y definidas al comienzo del mismo.

Así los procesos de la estructura principal de MÉTRICA Versión 3 son:

- Planificación de Sistemas de Información (PSI).
- Estudio de Viabilidad del Sistema (EVS).
- Desarrollo de Sistemas de Información (DSI).

4.2.3. PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN (PSI)

El proceso de Desarrollo de MÉTRICA Versión 3 recoge todas las actividades y tareas que se deben llevar a cabo para desarrollar un sistema, cubriendo desde el análisis de requisitos hasta la instalación del software. Además de las tareas relativas al análisis, incluye dos partes en el diseño de sistemas: arquitectónico y detallado. También cubre las pruebas unitarias y de integración del sistema, aunque siguiendo la norma ISO 12.207 no propone ninguna técnica específica y destaca la importancia de la evolución de los requisitos del sistema que se va a diseñar.

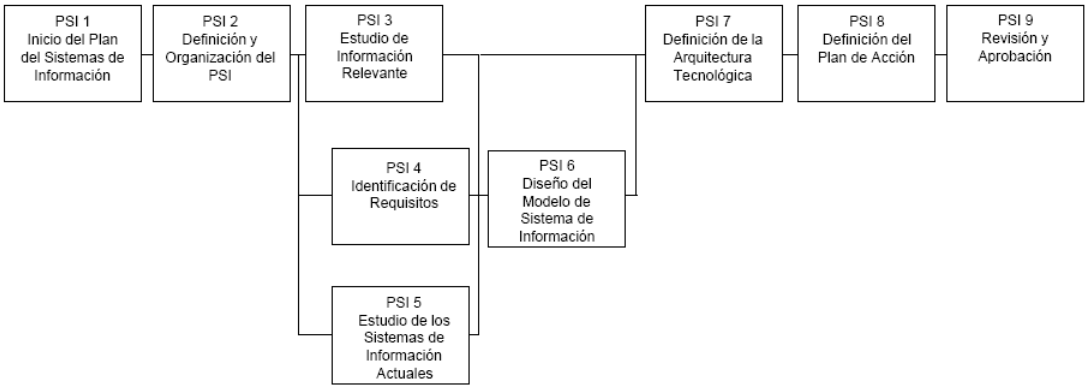


Figura 16. Planificación del Sistema de Información.

Este proceso es, sin duda, el más importante de los identificados en el ciclo de vida de un sistema y se relaciona con todos los demás. Las actividades y tareas propuestas por la norma se encuentran más en la línea de un desarrollo clásico, separando datos y procesos, que en la de un enfoque orientado a objetos.

4.2.4. ESTUDIO DE VIABILIDAD DEL SISTEMA (EVS)

El propósito de este proceso es analizar un conjunto concreto de necesidades, con la idea de proponer una solución a corto plazo. Los criterios con los que se hace esta propuesta no serán estratégicos sino tácticos y relacionados con aspectos económicos, técnicos, legales y operativos. Los resultados del Estudio de Viabilidad del Sistema constituirán la base para tomar la decisión de seguir adelante o abandonar. Si se decide seguir adelante pueden surgir uno o varios proyectos que afecten a uno o varios sistemas de información. Dichos sistemas se desarrollarán según el resultado obtenido en el estudio de viabilidad y teniendo en cuenta la cartera de proyectos para la estrategia de implantación del sistema global.

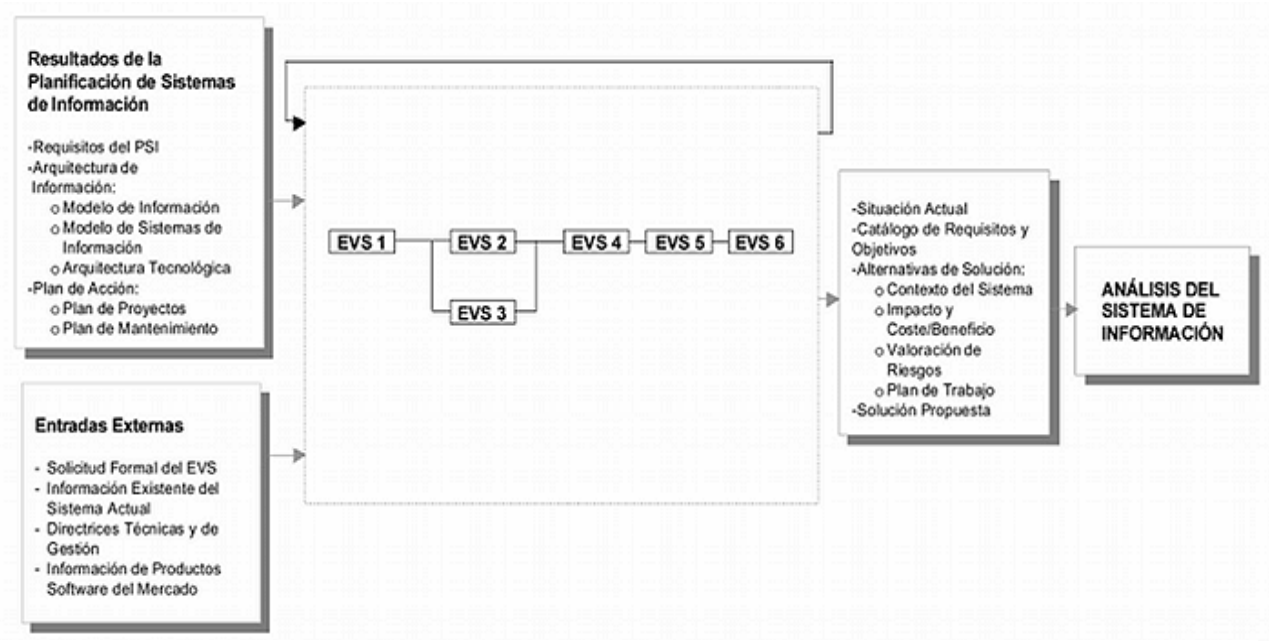


Figura 17. Fases del Estudio de Viabilidad del Sistema.

Se ha considerado que este proceso es obligatorio, aunque el nivel de profundidad con el que se lleve a cabo dependerá de cada caso. La conveniencia de la realización del estudio de la situación actual depende del valor añadido previsto para la especificación de requisitos y para el planteamiento de alternativas de solución. En las alternativas se considerarán soluciones "a medida", soluciones basadas en la adquisición de productos software del mercado o no y soluciones mixtas. Para valorar las alternativas planteadas y determinar una única solución, se estudiará el impacto en la organización de cada una de ellas, la inversión y los riesgos asociados.

El resultado final de este proceso son los productos relacionados con la solución que se propone para cubrir las necesidades concretas que se plantearon en el

4.2.5. DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI)

El propósito del Diseño del Sistema de Información (DSI) es obtener la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información. A partir de dicha información, se generan todas las especificaciones de construcción relativas al propio sistema, así como la especificación técnica del plan de pruebas, la definición de los requisitos de implantación y el diseño de los procedimientos de migración y carga inicial, éstos últimos cuando proceda.

El diseño de la arquitectura del sistema dependerá en gran medida de las características de la instalación, de modo que se ha de tener en cuenta una participación activa de los responsables de Sistemas y Explotación de las Organizaciones para las que se desarrolla el sistema de información. Este proceso consta de un primer bloque de actividades, que se realizan en paralelo, y cuyo objetivo es obtener el diseño de detalle del sistema de información que comprende la partición física del sistema de información, independiente de un entorno tecnológico concreto, la organización en subsistemas de diseño, la especificación del entorno tecnológico sobre el que se despliegan dichos subsistemas y la definición de los requisitos de operación, administración del sistema, seguridad y control de acceso.

Al igual que en el proceso de Análisis del Sistema de Información (ASI), antes de proceder a la especificación de los componentes, se realiza una verificación y validación, con objeto de analizar la consistencia entre los distintos modelos y formalizar la aceptación del diseño de la arquitectura del sistema por parte de los usuarios de Explotación y Sistemas.

4.2.6. INTERFACES DE MÉTRICA VERSIÓN 3

La estructura de MÉTRICA Versión 3 incluye también un conjunto de interfaces que definen una serie de actividades de tipo organizativo o de soporte al proceso de desarrollo y a los productos, que en el caso de existir en la organización se deberán aplicar para enriquecer o influir

en la ejecución de las actividades de los procesos principales de la metodología y que si no existen habrá que realizar para complementar y garantizar el éxito del proyecto desarrollado con MÉTRICA Versión 3.

La aplicación de MÉTRICA Versión 3 proporciona sistemas con calidad y seguridad, no obstante puede ser necesario en función de las características del sistema un refuerzo especial en estos aspectos, refuerzo que se obtendría aplicando la interfaz.

Las interfaces descritas en la metodología son:

- Gestión de Proyectos [28] (GP).
- Seguridad (SEG).
- Aseguramiento de la Calidad (CAL).
- Gestión de la Configuración (GC).

GESTIÓN DE PROYECTOS

La Gestión de Proyectos tiene como finalidad principal la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema de Información.

Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos lo más pronto posible, lo cual evitará desviaciones temporales y económicas.

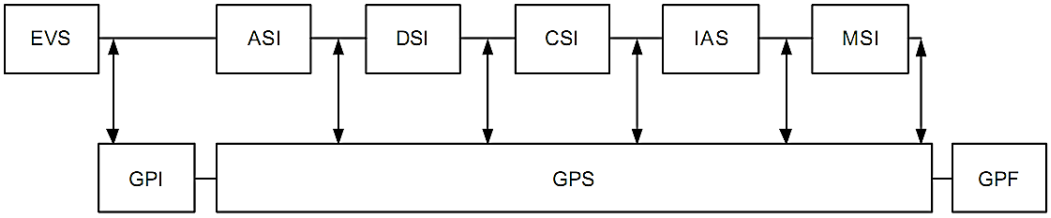


Figura 18. Fases de la Gestión de Proyectos.

La Interfaz de Gestión de Proyectos de MÉTRICA Versión 3 contempla proyectos de desarrollo de sistemas de información en sentido amplio, acorde con EUROMÉTODO[12], se consideran proyectos de desarrollo de nuevos sistemas de información y también los proyectos de ampliación y mejora de los ya existentes.

Las actividades de la Interfaz de Gestión de Proyectos son de tres tipos:

- **Actividades de Inicio del Proyecto (GPI)**: Permiten estimar el esfuerzo y establecer la planificación del proyecto.
- **Actividades de Seguimiento y Control (GPS):** Supervisan la realización de las tareas por parte del equipo de proyecto y gestionando las incidencias y cambios en los requisitos que puedan presentarse y afectar a la planificación del proyecto.
- **Actividades de Finalización del Proyecto**: Se procede al cierre y registro de la documentación de gestión.

Estas actividades pueden requerir, en función de la complejidad del proyecto, el soporte de herramientas comerciales de gestión de proyectos.

SEGURIDAD

El análisis de los riesgos constituye una pieza fundamental en el diseño y desarrollo de sistemas de información seguros. Si bien los riesgos que afectan a un sistema de información son de distinta índole: naturales (inundaciones, incendios, etc.) o lógicos (fallos propios, ataques externos, virus, etc.) son estos últimos los contemplados en la interfaz de Seguridad de MÉTRICA Versión 3.

Objetivo de la interfaz de Seguridad

El objetivo de la interfaz de seguridad de MÉTRICA Versión 3 es incorporar en los sistemas de información mecanismos de seguridad adicionales a los que se proponen en la propia metodología, asegurando el desarrollo de cualquier tipo de sistema a lo largo de los

[12] http://www.csi.map.es/csi/pdf/em_v1.pdf/. Último acceso 07-06-2011

procesos que se realicen para su obtención. La interfaz de Seguridad hace posible incorporar durante la fase de desarrollo las funciones y mecanismos que refuerzan la seguridad del nuevo sistema y del propio proceso de desarrollo, asegurando su consistencia y seguridad, completando el plan de seguridad vigente en la organización o desarrollándolo desde el principio

Las valoraciones sobre la seguridad deben realizarse en función de las características del sistema sin perder de vista además que, al ser finitos los recursos, no pueden asegurarse todos los aspectos del desarrollo de los sistemas de información, por lo que habrá que aceptar un determinado nivel de riesgo concentrándose en los aspectos más comprometidos o amenazados, que serán diferentes según las circunstancias.

GESTIÓN DE CONFIGURACIÓN

La interfaz de gestión de la configuración consiste en la aplicación de procedimientos administrativos y técnicos durante el desarrollo del sistema de información y su posterior mantenimiento. Su finalidad es identificar, definir, proporcionar información y controlar los cambios en la configuración del sistema, así como las modificaciones y versiones de los mismos. Este proceso permitirá conocer el estado de cada uno de los productos que se hayan definido como elementos de configuración, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan.

Objetivo del proceso de Gestión de Configuración

La interfaz de gestión de configuración de MÉTRICA Versión 3 permite definir las necesidades de gestión de configuración para cada sistema de información, recogiéndolas en un plan de gestión de configuración, en el que se especifican actividades de identificación y registro de productos, que se realizan durante todas las actividades de MÉTRICA Versión 3 asociadas al desarrollo y mantenimiento del sistema de información.

A su vez, permite controlar el sistema como producto global a lo largo de su creación, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores durante el mismo, lo que se traduce en un aumento de calidad del proceso de desarrollo y de mejora de la productividad en la organización.

La gestión de configuración facilita además el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y minimizando el tiempo de implementación de un cambio, tanto evolutivo como correctivo.

4.2.7. RESUMEN

Como resultado de la ejecución de cada uno de los procesos del ciclo de vida de Métrica 3 se han obtenido los siguientes entregables para el proyecto NQPL:

- Estudio de Viabilidad del Sistema (EVS).
- Análisis del Sistema de Información (ASI).
- Gestión del Proyecto (GP).
- Manual de Usuario (MU).

4.3. TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO DE NQPL

4.3.1. C#

INTRODUCCIÓN AL LENGUAJE C#

C# [29], pronunciado C sharp, es un lenguaje diseñado por Microsoft para combinar el potencial de C/C++ y la productividad de Visual Basic y estandarizado por la ECMA-334 [13] y la ISO/IEC 23270:2006. Las especificaciones iniciales de este lenguaje de programación revelan similitudes con Java que son obvias, incluyendo la sintaxis, la robusta integración web y la gestión automatizada de memoria.

CARACTERÍSTICAS GENERALES

Entre las características que se presentan a continuación algunas son propias de la plataforma .NET más que del propio lenguaje Orientación a objetos: C# es un lenguaje de programación orientado a objetos puro, es decir, no admite funciones ni variables globales, todo el código debe definirse dentro de tipos de datos (clases, estructuras...). Esto reduce los conflictos de nombres y facilita la legibilidad del código.

Orientado a objetos: C# soporta las características principales del paradigma de la programación orientada a objetos:

- Encapsulación: Además de los modificadores *public*, *private* y *protected* se añade uno nuevo: *internal*. Internal indica que el elemento sólo podrá ser accedido desde su propio ensamblado.
- Herencia: C#, al igual que Java, sólo soporta herencia simple debido a las ambigüedades que produce la herencia múltiple. La diferencia está en que para C# todos los métodos serán sellados y para poder redefinir alguno habrá que usar

[13] http://www.ecma-international.org/publications/standards/Ecma-334.htm. Último acceso 07-06-2011

el modificador virtual (como en C++). Con esto las llamadas a los métodos serán más rápidas, ya que si no está presente el modificador virtual se tomará la llamada al método de la clase base.

- Polimorfismo: La herencia en el lenguaje C# permite que clases derivadas empleen un mismo método diferente heredado.

Seguridad de tipos: C# es un lenguaje fuertemente tipado, esto evita cometer errores que son difíciles de detectar.

- Sólo se pueden realizar conversiones entre tipos que sean compatibles.
- No se pueden usar variables sin inicializar, ya que no tendrían un estado inicial seguro.
- En los accesos a elementos de tablas (arrays, matrices,…) se comprueba que los índices no se salgan de rango.
- Se informa con excepciones cuando se producen desbordamientos en operaciones aritméticas.
- Aunque un método posea un número indeterminado de parámetros de un tipo, se comprueban.

Sistema de tipos unificado: En C# todos los tipos de datos que se definan deben derivar de una clase base común llamada *System.Object*. Esto facilita la creación de colecciones genéricas ya que pueden almacenar objetos de cualquier tipo.

Sencillez: En C# se eliminan elementos incluidos en otros lenguajes y que son innecesarios en .NET, como por ejemplo:

- No necesita ficheros adicionales al código fuente como son los ficheros de cabecera o los ficheros IDL ya que el código escrito es auto contenido.
- El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina utilizada; esto fomenta la portabilidad. Es una de las mejoras con respecto a C++.
- No se incluyen elementos como las macros, herencia múltiple de otros lenguajes.

Modernidad: En C# se han incluido elementos nuevos que a lo largo de los años se ha comprobado que son muy útiles para ciertas aplicaciones y que en otros lenguajes como Java y C++ hay que simularlos. Por ejemplo:

- Creación del tipo básico decimal para permitir realizar operaciones de alta precisión con reales de 128 bits.
- Creación de la instrucción *foreach* para recorrer colecciones con facilidad y que además se puede ampliar para tipos definidos por el usuario.
- Creación del tipo básico *string* para la representación de cadenas.
- Orientación a componentes: se incluyen en la sintaxis elementos del diseño de componentes. Con lo cual se pueden definir de una forma sencilla propiedades, eventos o atributos.

Gestión automática de memoria: C# dispone de recolector de basura, lo cual evita al programador el tener que preocuparse de destruir objetos, liberar memoria. Sin embargo, también se dispone de un método manual para liberar recursos, usando la instrucción *using*.

Instrucciones seguras: Se evitan errores comunes cometidos en otros lenguajes imponiendo una serie de restricciones en algunas instrucciones, como obligar que todos los casos de un *switch* terminen con un *break* o un *goto*.

Extensibilidad de tipos básicos: A través de estructuras se permite definir tipos de datos que tengan las mismas optimizaciones que los tipos de datos básicos, esto es, que se puedan almacenar en pila directamente.

Extensibilidad de operadores: Al igual que C++ y a diferencia de Java, se puede redefinir el significado por defecto de gran parte de los operadores.

Eficiencia: En C# el código incluye muchas restricciones por seguridad, como no permitir el uso de punteros; sin embargo, a través del modificador *unsafe* se pueden crear unas regiones de código inseguras donde saltarse dichas restricciones.

Compatibilidad: Se permite incluir en códigos escritos en C# fragmentos de códigos escritos en otros lenguajes, así como acceder a funciones sueltas no orientadas a objetos (entendiendo orientado a objetos en el sentido de ser puro, es decir de que todo código debe estar definido en un tipo de datos) como las DLLs. Esto es posible debido a la característica anterior y a los Platform Invocation Services (PInvoke) que nos ofrece el CLR.

Soporte de XML: El compilador de C# es capaz de generar la documentación en XML a partir de los comentarios que el programador haya escrito en su código fuente. Esto es de gran ayuda puesto que código y documentación estarán en el mismo documento. Para generar la documentación XML se debe usar la opción de compilación /doc y habrá que darle el nombre que va a tener dicho documento. Esto se explicará más adelante al tratar el compilador.

Al crearse la documentación XML se usan una serie de etiquetas que forman un conjunto inteligible para otras herramientas de Visual.NET.

Espacio de nombres: Es un novedad de C# que ayuda a tener ordenados los tipos de datos. Se podría comparar con los ficheros que se organizan por directorios. Una de las ventajas más obvias es que se pueden declarar dos clases distintas con el mismo nombre sin más que hacerlas pertenecer a espacios de nombres diferentes.

LA ELECCIÓN DE C#. CONCLUSIONES

Visual Studio admite el lenguaje C# con un editor para código completo, asistentes de depuración para dicho código fácil de usar y plantillas para proyectos entre otras herramientas por lo que la implementación resultará en cierto modo sencilla y ágil. Además, las bibliotecas de clases .NET proporcionan un acceso servicios del sistema operativo y otras clases que están perfectamente diseñadas para acelerar el ciclo de desarrollo de forma significativa.

A continuación se muestra un esquema del Entorno de Desarrollo Integrado (IDE) de Visual C# donde se puede apreciar la estructura de la interfaz de desarrollo sobre la cual se va a implementar el proyecto.

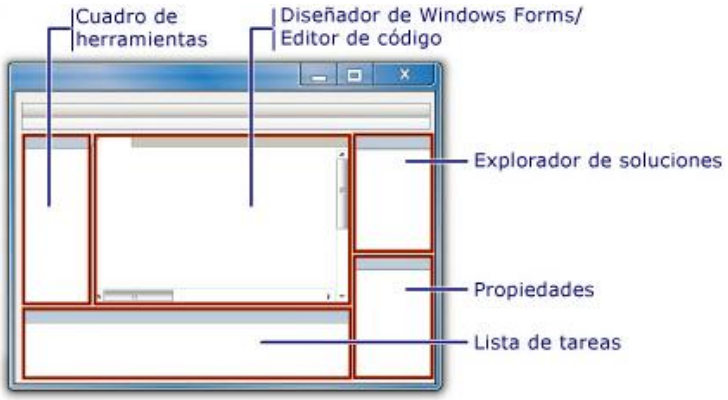


Figura 19. Interfaz Visual Studio.

4.3.2. XAML

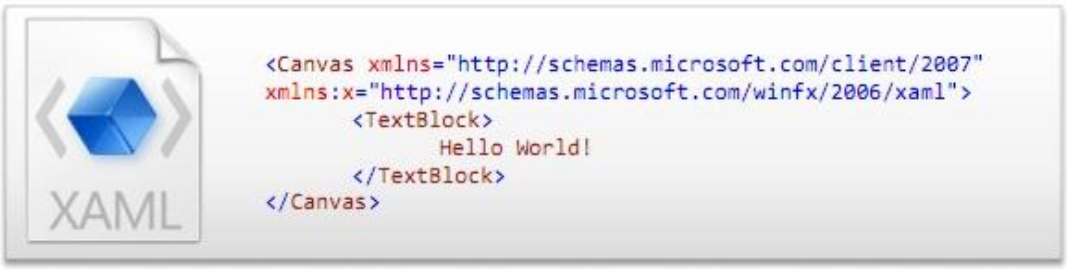
QUÉ ES XAML

XAML es un lenguaje declarativo de marcado, tal y como se aplica en el modelo de programación .NET Framework, que simplifica la creación de la interfaz de aplicaciones. Estrictamente hablando, XAML es un lenguaje de marcado que sigue las reglas sintácticas de XML, donde cada elemento tiene un nombre y puede tener varios atributos. Los elementos se definen uno dentro de otro formando una estructura arbórea, lo que le da expresividad y semántica a este lenguaje en su correspondencia con un marco de trabajo como lo es WPF y .NET.

Se pueden crear elementos visibles de la UI en el marcado XAML declarativo y, a continuación, separar la definición de la UI de la lógica en tiempo de ejecución mediante archivos de código subyacente, que se unen al marcado mediante definiciones de clases parciales. Representa a su vez la creación directa de instancias de objetos en un conjunto concreto de tipos de respaldo definido en ensamblados lo que lo diferencia mucho de otros tipos de lenguajes de marcado. El código XAML habilita un flujo de trabajo en el que las

partes independientes pueden funcionar en la UI y en la lógica de una aplicación, a través de herramientas potencialmente diferentes.

En el ejemplo siguiente se muestra cómo podría crear un botón como parte de una



UI. Este ejemplo se ha pensado simplemente para que se tenga una idea de cómo representa el código XAML las metáforas de programación de la UI común.

Esencialmente, con XAML expresaremos declarativamente la creación de estructuras arbóreas de objetos .NET. Con WPF como marco de trabajo, estos objetos podrán ser desplegados visualmente y así los usuarios podrán interactuar con ellos de forma directa, permitiendo conformar una interfaz de aplicación rica y fácil de interpretar. A su vez, estos

Figura 20. Ejemplo de código XAML.

objetos visuales podrán ser conectados con el resto de la lógica de la aplicación por muy compleja que sea consiguiendo así que XAML y WPF conformen la pareja perfecta para el desarrollo de este tipo de aplicaciones pero conservando su independencia.

QUÉ APORTA XAML

Por su naturaleza declarativa, XAML es apropiado para especificar la interfaz de usuario de una aplicación de modo separado de la lógica de negocio propia de la aplicación. En este sentido, la combinación de los tres ingredientes: lenguaje de programación .NET (C# en este proyecto), XAML y WPF, nos ofrece un adecuado soporte para desarrollar aplicaciones con la arquitectura conocida como MVVC, ModeloVista-Vista-Controlador (*ver "7.1. DESCRIPCIÓN DE LA SOLUCIÓN"),* que no es más que una readaptación del modelo MVC (Modelo-Vista-Controlador). Esta arquitectura, por su estructura de capas, nos

permite lograr mayor eficiencia de desarrollo y mayor flexibilidad del producto de cara al usuario final. Por ejemplo, si estuviéramos desarrollando una aplicación para gestionar un sistema que albergue información de usuarios, con XAML y WPF se podría expresar cómo visualizar e interactuar con la información almacenada, pudiendo por ejemplo presentar animaciones, carruseles interactivos, etc, y a su vez conectar ese capa visual con una base de datos gestionada mediante código C# y SQL. Con esto podremos conseguir una estrecha relación de interacción cómoda y entre el usuario, C#, XAML, WPF, .NET y SQL.

La integración de C#, XAML y WPF, puede llevarse a cabo de manera cómoda y estructurada con la herramienta Microsoft Visual Studio, por lo que ésta ha sido la elegida para desarrollar el proceso de reingeniería para la migración del software mostrado en el presente Proyecto Fin da Carrera.

4.3.3. WINDOWS PRESENTATION FUNDATION (WPF)

INTRODUCCIÓN AL MARCO DE TRABAJO DE WPF

A pesar de la riqueza interactiva de las aplicaciones clientes, introducidas desde los orígenes del propio Windows, y que han llegado a su máxima expresión con la tecnología .NET y con los recursos de Windows Forms, lo cierto es que bajo las pretensiones integradoras y abarcadoras de las aplicaciones de hoy en día, estas capacidades tienen limitaciones. No es simple integrar en nuestras aplicaciones el despliegue de documentos, dar efectos tridimensionales, o colocar imágenes, audio y vídeo. Posiblemente algunas de las atractivas presentaciones que veamos por ahí, han tenido que desarrollarse bajo el embrollo de mezclar variadas tecnologías y formatos (controles ActiveX, Flash, PDF, etc.) exigiendo de varios desarrolladores, y requiriendo de ellos un alto nivel de complejidad para coordinar sus respectivos trabajos y resultados.

En ocasiones, nos podemos encontrar ante la necesidad de mejorar la apariencia de una aplicación desarrollada por otros. O nuestros clientes requieren la mejora de la interactividad de una aplicación desarrollada por nosotros o incluso se nos plantea el inconveniente de tener que ser al mismo tiempo un buen diseñador artístico y un buen

programador. Con PowerPoint uno es capaz de preparar y retocar las presentaciones logrando atractivos efectos y apariencia. Esto mismo se desearía hacer en muchas ocasiones con las propias aplicaciones y que además incorporasen la funcionalidad y la lógica de nuestro negocio, pudiendo retocar ambas partes por separado. WPF es la propuesta de Microsoft para lograr todo esto.

LA DECISIÓN DE EMPLEAR WPF

WPF es toda una plataforma que da soporte para poder colocar en nuestras aplicaciones una amplia riqueza visual e interactiva, e integrarlas fácilmente con la lógica del negocio. Los elementos visuales de nuestras aplicaciones podrán tener ahora transparencia, brillo, tonalidades, sombra, reflejo, efectos tridimensionales y animaciones. Se podrá desplegar documentos y colocar en las propias aplicaciones audio y vídeo. Las aplicaciones podrán tener capacidad de enlace y navegación basado en el modelo de páginas de la Web. La reutilización será propiciada gracias a la posibilidad de definir estilos propios y plantillas, aumentando la productividad de desarrollo y dando una apariencia personalizada a las aplicaciones. Además la arquitectura abierta de WPF nos permite definir e implementar nuestros propios controles y componentes personalizados.

La potencialidad gráfica de WPF se basa a su vez en la tecnología DirectX de Microsoft que sabe aprovechar al máximo las posibilidades gráficas de que disponga el hardware donde se ejecute la aplicación.

Las capacidades de WPF son totalmente asequibles desde cualquier lenguaje .NET y utilizables programáticamente en el modo tradicional de la programación orientada a objetos. No obstante Microsoft nos propone, junto a WPF, un lenguaje declarativo XAML que se acopla a la perfección con WPF.

4.3.4. COMBINACIÓN DE XAML, WPF Y .NET

ANÁLISIS DE LA PROPUESTA

Por lo general, todo elemento en XAML se corresponde con una clase de .NET, en particular de WPF, y todo atributo XAML se corresponde con el nombre de una propiedad, o de un evento, de dicha clase. A la inversa, aunque no todas las clases del CLR se puedan representar en XAML, aquellas que tengan un constructor por defecto y propiedades públicas pueden ser instanciadas declarativamente en código escrito en XAML si existiesen los convertidores apropiados para convertir una cadena (que es en definitiva el único tipo de valor que se escribe en un lenguaje como XAML) en el tipo de .NET correspondiente (numérico, booleano, enumerado, etc.).

COMBINACIÓN DE XAML, WPF Y C#

Lo que hace más valioso a la integración entre WPF y XAML es que al quedar la interfaz de usuario especificada con un lenguaje de marcas y declarativo, la hace legible y modificable por el usuario, y lo que es más importante, fácilmente susceptible de ser analizada y procesada por herramientas. Esto ofrece a terceros la posibilidad de crear herramientas de diseño visual que soporten XAML. O lo que para este proyecto es más importante aún, la creación de controles personalizados que aporten una funcionalidad concreta a dichos controles, y la posibilidad de definir estilos propios para aportar a nuestra presentación la apariencia deseada. La separación entre código XAML para expresar la interfaz y apariencia, y código de un lenguaje .NET como C# para expresar la lógica de la aplicación, facilita que los diseñadores con capacidades artísticas (posiblemente poco habilidosos para la programación) puedan integrar mejor y con más efectividad su trabajo con los programadores concentrados en la lógica de la aplicación (a veces poco talentosos en términos de diseñar una atractiva apariencia).

XAML es extensible, como su propio nombre lo indica, lo que significa que los desarrolladores podrán crear sus propios controles y elementos personalizados. Este hecho ha sido fundamental para la consecución de este proyecto, ya que gran parte de los objetos a migrar no están definidos en WPF, y muchos otros necesitan ser tratados de un modo

especial para su acceso a bases de datos y control de eventos, por lo que se necesita poder definirlos como controles personalizados.

En la figura 19, se puede observar cómo al crear un objeto en WPF mediante drag 'n drop desde la barra de herramientas de objetos al área de trabajo, en este caso un button, se genera automáticamente el código XAML asociado a ese objeto con las propiedades fundamentales definidas a la hora de crearlo.

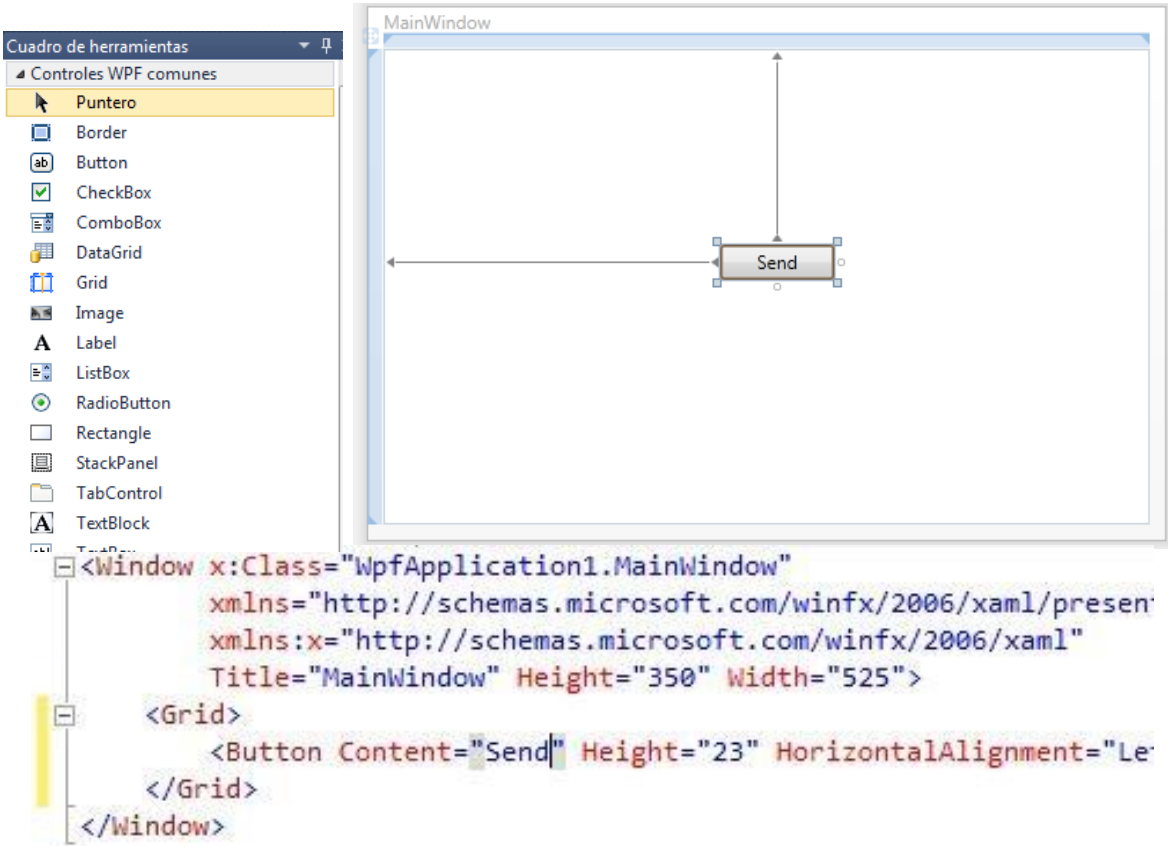


Figura 21. Interfaz desarrollo Windows Presentation Foundation.

XAML Y DATABINDING

En este apartado se presenta lo que aporta la tecnología DataBinding a las aplicaciones en WPF. Como se ha mencionado anteriormente, representa una interfaz de comunicaciones o enlace a datos por medio del patrón Observer en su versión más moderna: el evento entre dos propiedades que forman parte de la definición de las dos clases involucradas. Cuando el usuario realiza operaciones sobre los controles visuales de

las aplicaciones, dichas acciones se traducen a eventos que generan peticiones de actualización visual.

Este sistema mantiene la aplicación en un estado persistente sin que el programador, desarrollador o diseñador tengan que realizar tareas más que la propia integración del enlace a los datos.

Los enlaces a datos pueden tener direcciones en cuanto a los objetos que requieren de la actualización, habiendo tres formas posibles de actualización:

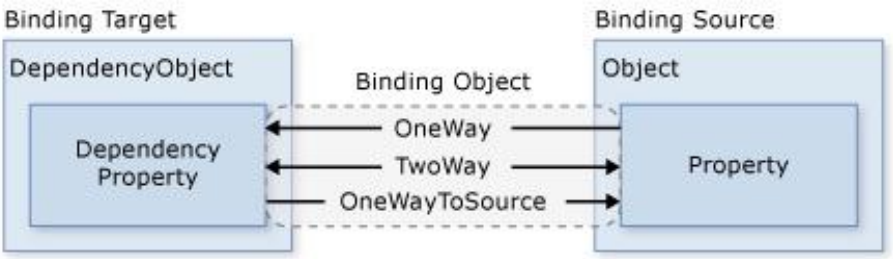


Figura 22. Flujo de datos del bindado.

- **Directa (OneWay):** EL objeto destino es actualizado por los datos que se encuentran en el objeto origen.
- **Inversa (OneWayToSource):** El objeto origen es actualizado por los datos que se encuentran en el objeto destino.
- **Completa (TwoWay):** Tanto el origen como el destino mantienen el estado persistente de los datos en una única transacción de comunicaciones.

WPF proporciona una manera sencilla y potente de auto-actualizar datos entre el modelo de negocio y la interfaz de usuario. Este mecanismo se denomina DataBinging. Cada vez que los datos del modelo de negocio cambian, automáticamente refleja estos cambios en la interfaz de usuario y viceversa. Este es el método preferido de WPF de mostrar información en la interfaz de usuario.

El origen de un DataBinding puede ser una propiedad .NET o una DependencyProperty mientras que el destino debe ser una DependencyProperty. Para que

el bindado funcione correctamente en ambos extremos, el bindado debe proporcionar una notificación de que se ha producido un cambio y decirle al bindado cuándo debe actualizar el valor destino. Esta tarea suele ser realizada por el evento *PropertyChanged* de la interfaz *INotifyPropertyChanged*.

El DataBinding suele ser implementado en XAML utilizando la etiqueta de marcado *{Binding Propiedad, IdControl, Trigger}.* En el siguiente ejemplo se muestra un bindado simple entre el texto de un TextBox y una Label que refleja el valor escrito en el TextBox:

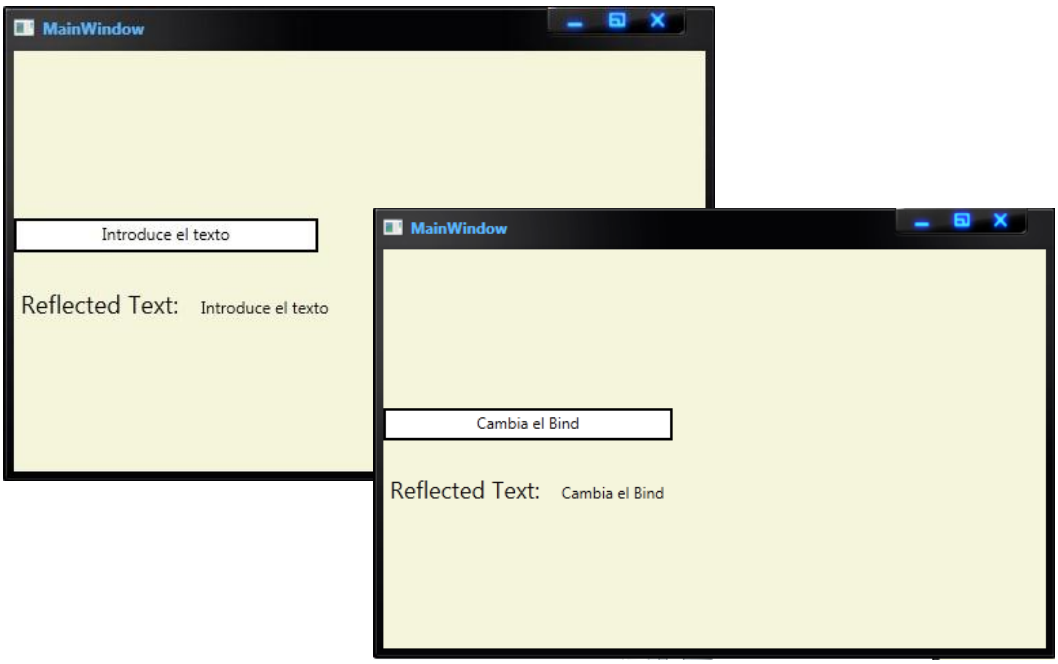


Figura 23. Binding WPF.

Como se puede observar, al modificar el valor del TextBox, se va actualizando el valor mostrado en el Label. Esto se debe al código Xaml que tiene el Label y en el que se incluye el bindado:

```
<Label Content="{Binding Text, ElementName=txtInput,
       UpdateSourceTrigger=PropertyChanged}" Width="153"
       HorizontalAlignment="Left" Margin="133,175,0,97"/>
```

Figura 24. Binding XAML.

PropertyChanged de la propiedad *Text* del TextBox *txtInput*, que le asigne a su propiedad Content el valor de esa propiedad bindada del TextBox *Text*.

La funcionalidad básica que ofrece los enlaces a datos son los siguientes:

- Conversión de datos en la capa gráfica (View) para mostrar los datos origen en la interfaz gráfica de una manera distinta. Enlace a colecciones de datos en orígenes diversos como bases de datos, ficheros planos, ficheros Excel bajo un entorno unificado.
- Inclusión de datos en plantillas para su reutilización y abstracción de modelos gráficos.
- Validación de datos previa a la realización de acciones por parte del usuario.
- Mecanismo de depuración para poder recibir el estado de las comunicaciones del enlace de datos.

Así, los controles contenidos, tales como botones y controles contenedores como un DataGrid han mejorado su funcionalidad para permitir la presentación de objetos de datos o colecciones de datos. En resumen, significa un vínculo de unión entre la capa del modelo y la de la vista en tiempo de ejecución. Un ejemplo de Databinding en este proyecto estaría representado por el binding existente entre los datos que se obtienen de una consulta a la base de datos y el DataGrid donde se muestran:

```
<DataGrid Name="QueryDataGrid" Grid.Row="0" Visibility="Collapsed" ItemsSource="Binding"
          AutoGenerateColumns="true" Height="Auto" MaxHeight="200"
          Background="White" VerticalAlignment="Top" />


  // Creates a binding source
  BindingSource bSource = new BindingSource();
  //Set the BindingSource DataSource so we can link DataSource info to DataGrid ADO control.
  bSource.DataSource = oDataTable;
  //Set the DataGridView DataSource. This shows bSour
  oDataView.ItemsSource = bSource;
  DataTable = oDataTable;
```

CUSTOMER	CONTACT_FIRST	PHONE_NO
Signature Design	Dale J.	(619) 530-2710
Dallas Technologies	Glen	(214) 960-2233
Buttle, Griffith and Co.	James	(617) 488-1864
Mrs. Beauvais		
Anini Vacation Rentals	Leilani	(808) 835-7605

Figura 25. Databinding en XAML.

LINQ

Language Integrated Query es un complemento que integró Visual Studio 2008 y .NET framework 3.5 y que acerca más el mundo de los datos a los objetos. Este lenguaje alberga sentencias nativas parecidas a las de SQL y puede actuar sobre cualquier fuente de datos, expresando consultas eficientemente sobre objetos utilizados en .NET, como por ejemplo un *IEnumerable*:

```
IEnumerable<string> oDistinct;
            oDistinct = from x in oOutput
                        where (x.ToLower() == "X") || (x.ToLower() == "Y") ||
(x.ToLower() == "J") || (x.ToLower() == "K")
                        select x;
```

En este ejemplo, LINQ lanza una consulta sobre el objeto *oOutput* y devuelve otro objeto del mismo tipo que contendrá todos los elementos que cumplan la condición estipulada en la sentencia *where*, es decir, aquellos que valgan X,Y,J o K.

PROBLEMÁTICA DE LA FUSIÓN E INTEGRACIÓN

El problema principal que se presenta es llevar acabo la integración de los 3 componentes de programación necesarios para implementar la solución. Como ya se ha comentado antes, se necesita un componente que permita a XAML. C# y WPF comunicarse siguiendo un modelo MVC en el que cada uno tiene bien definida su función, pero su trabajo no es inherente por completo al resto de capas.

A su vez, existe un problema a la hora de integrar esta solución como componente de Office, qué librerías nos van a permitir establecer la comunicación entre objetos desde la aplicación y Office y entre Visual Studio y la base de datos. Habitualmente es necesario un proveedor de objetos que permitan almacenar información de la base de datos y pasárselos a Visual Studio para, mediante clases, utilizarla y mostrársela al usuario.

En cuanto a la comunicación entre Office y Visual Studio, estaríamos hablando de objetos COM que mediante las librerías que ofrece Interop, pueden ser manejados desde

Visual Studio, por lo que no supondría ningún problema de integración de componentes. Se resume con más detalle en el siguiente apartado.

SOLUCIÓN A LA PROBLEMÁTICA

Para solucionar el problema de integración de las técnicas que se desean emplear para implementar la aplicación y asegurarnos de que existirá una comunicación adecuada entre la base de datos y nuestro sistema, se presentan las siguientes alternativas:

- **Visual Studio .NET:** Soluciona la integración de C#, XAML y WPF.
- **Proveedores de datos**: Un proveedor de datos de .NET Framework sirve para conectarse a una base de datos, ejecutar comandos y recuperar resultados. Esos resultados se procesan directamente o se colocan en un DataSet de ADO.NET u otro driver de comunicación, con el fin de exponerlos al usuario para un propósito específico. EL framework de .NET pone a disposición de los desarrolladores los siguientes proveedores de datos:

  - ADO.NET: Conjunto de componentes de software que permiten acceder a datos y servicios de datos. En el IDE Visual Studio .NET, existe la funcionalidad para crear las subclases especializadas de las clases del DataSet para un esquema particular de base de datos, permitiendo el acceso conveniente a cada campo a través de propiedades fuertemente tipadas.

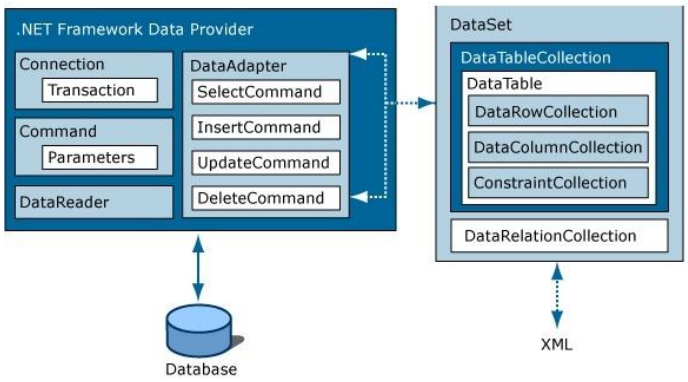


Figura 26. Arquitectura ADO.NET. y estructuras de datos.

- OLEDB: Permite el acceso a datos separándolos de la aplicación.

- ODBC: Permite acceder a cualquier información de la base de datos sin importar el GSBD que se esté utilizando. Esto es posible gracias a una capa intermedia denominada Interfaz de Cliente (*del inglés CLient Interface, CLI*) traduce las consultas para que el SGBD las entienda.

4.3.5. SISTEMA OPERATIVO WINDOWS

El entorno en el que ha sido desarrollado el proceso de implementación de NQPL ha sido el sistema operativo Windows. La empresa que cuenta con la herramienta que se permite trabajar con gran cantidad de librerías e instrumentos pertenecientes a Microsoft que permiten combinar todas las técnicas previamente expuestas. En este sentido, se ha desechado la posibilidad de utilización de otros sistemas operativos para el desarrollo de la migración.

Aparte de ello, es abrumadora la diferencia entre el uso del sistema operativo Windows y los demás sistemas operativos, por lo que el uso de esta plataforma resulta aún más evidente.

Como puede observarse, el uso y popularidad de Windows frente a los al resto de sistemas operáticos es muy superior.

Figura 27. Comparativa popularidad entre sistemas operativos.

Además de esto, el volumen de software desarrollado para Windows es muy superior al resto, aunque existe un notable detrimento y se puede observar que poco a poco el desarrollo OpenSource va abriéndose camino.

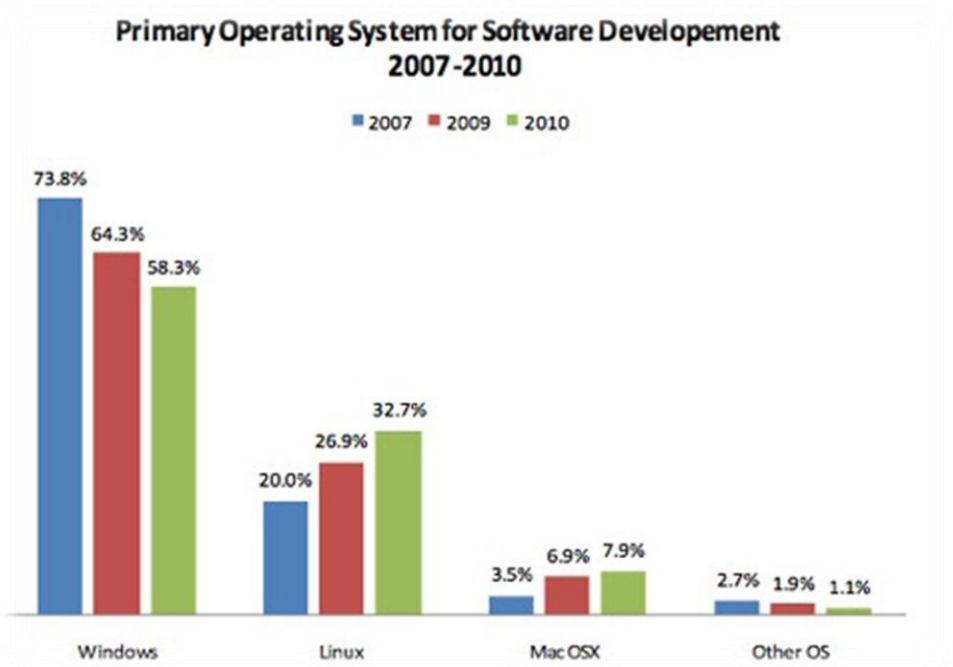


Figura 28. Comparativa desarrollo aplicaciones entre sistemas operativos.

4.4. HERRAMIENTAS EMPLEADAS EN EL DESARROLLO

4.4.1. MICROSOFT VISUAL STUDIO 2008-2010

INTRODUCCIÓN A VISUAL

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Posibilita a los desarrolladores a crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Además proporciona una gran compatibilidad para administrar no sólo el ciclo de vida del desarrollo de software completo pero también la interacción con los usuarios finales y los administradores de aplicaciones de empresa.

DECISIÓN

Al tratar las necesidades de una variedad de clientes muy amplia, de los programadores más pequeños e independientes, a los clientes de empresa más grandes, Visual Studio ofrece la oportunidad de crear soluciones de alta calidad independientemente del tamaño del proyecto o equipo, facilitando la integración de aplicaciones como elemento en los desarrollos de proyectos.

Microsoft Visual Studio hace cada día realidad su visión de crear aplicaciones Smart Clients, permitiéndole a los desarrolladores crear rápidamente aplicaciones de alta calidad conectadas con sistemas ya existentes, sin importar la plataforma sobre la cual estén

creadas, mejorando su experiencia. Desde el punto de vista de negocio se podrá contar con aplicaciones amigables para el usuario final de forma rápida y que permitirá tomar decisiones a corto plazo y con bajos costes.

En cuanto a la experiencia del usuario brinda mejoras en las siguientes áreas:

- **Desarrollo para Windows y .NET Framework 3.x:** Los desarrolladores podrán fácilmente contar con nuevas plataformas tecnológicas y entregar más funcionalidades a los usuarios fácilmente incorporando las nuevas características de Windows Presentation Foundation.
- **Desarrollo de aplicaciones para Office**: Visual Studio Tools for Office está ahora totalmente integrado con Visual Studio 2008 Professional. VSTO permite a los desarrolladores personalizar varias aplicaciones de Office, como Office y PowerPoint, mejorando la productividad del usuario y mejorando notablemente su despliegue.
- **Fácil manejo de datos**: Con la introducción de Language Integrated Query (LINQ) y otras nuevas características de acceso a datos, los desarrolladores pueden ahora manipular datos usando un acercamiento programático constante, es decir, emplear lenguaje de consulta sobre el propio lenguaje utilizado en capa del controlador. Véase un ejemplo en la siguiente figura:

```
private void cmdAlta_Click(object sender, EventArgs e)
{
// Damos de alta el registro e inicializamos
      LinqTest obj = new LinqTest();
      obj.AddUsuario( txtUsuario.Text ,
                  txtClave.Text,
                  dtpFxAlta.Value);
      PonerDatosEnPantalla(null);
}
```

Figura 29. Ejemplo código LINQ en Visual Studio 2008.

- **Nuevas experiencias en la Web**: Más allá de la infraestructura segura, confiable y extensible del IIS 7, los desarrolladores pueden crear fácilmente aplicaciones Web más interactivas, una ejecución más responsiva y más eficiente del lado del cliente usando la integración y el modelo de programación de ASP.NET, AJAX y otras extensiones y nuevas características incluyendo Silverlight.
- **Mejora la Administración del ciclo de vida de las aplicaciones (ALM)**: ALM proporciona gran ayuda, no sólo para administrar el ciclo de vida entero del desarrollo del software sino también para la interacción crítica de los usuarios finales y los gerentes de proyectos de aplicaciones empresariales.

VISUAL SOURCESAFE. SEGURIDAD Y CONTROL DE VERSIONES

En todo proyecto de desarrollo de software es necesario tener un control de versiones que controle las modificaciones que se llevan a cabo sobre el código de archivos de aplicación, evitar que un mismo fichero fuente esté siendo modificado, tener un archivo en check-out, simultáneamente por dos programadores y se pierda codificación al guardar cambios, hacer check-in, uno sobre el fichero del otro, así como de saber quién tiene en check-out cada fichero y quién ha realizado el último check-in de otros en cada momento. Esto evita futuros problemas de quién ha modificado qué y por qué y la tediosa tarea de tener que hacer un backtracking a versiones anteriores de archivos para que la aplicación vuelva a compilar y funcionar correctamente.

Visual SourceSafe es la respuesta a dichos problemas ya que se trata de un sistema de control de versiones a nivel de archivos que permite a muchos tipos de organizaciones trabajar en distintas versiones de un proyecto al mismo tiempo. Esta funcionalidad es especialmente ventajosa en un entorno de desarrollo de software, donde se usa para mantener versiones de código paralelas. Sin embargo, el producto también se puede utilizar para mantener archivos en cualquier otro tipo de equipo.

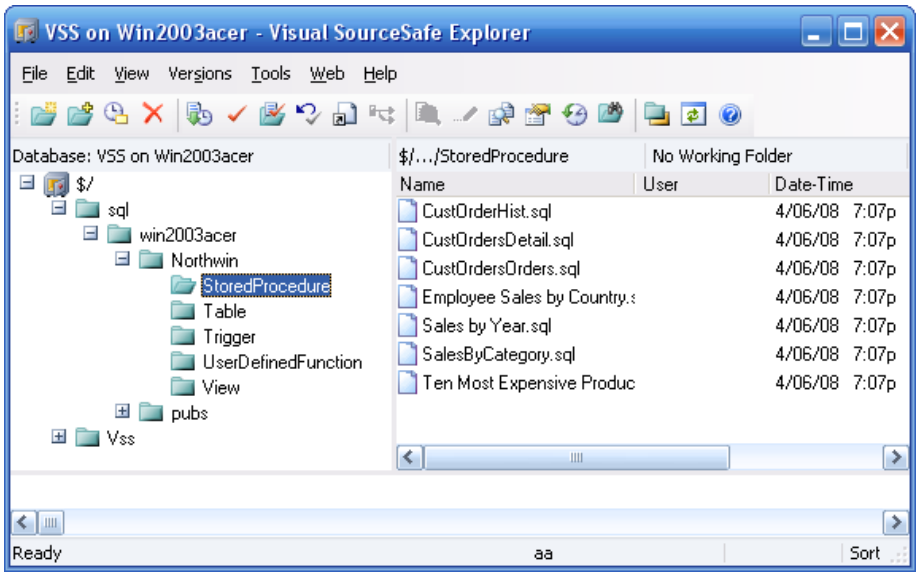


Figura 30.Interfaz de ejecución de SourceSafe.

4.4.2. SISTEMAS GESTORES DE DATOS BASADOS EN SQL

Como se ha mencionado a lo largo de todo el Capítulo II, SQL es el lenguaje declarativo de manipulación de datos más popular. La razón de su fama frente a los procedimentales se debe en gran parte a que es declarativo: solo hay que decir que se quiere hacer y no hay que especificar cómo hacerlo.

Pero la razón realmente importante que decanta la balanza hacia SGBD basados en SQL frente al resto no basados en SQL es porque en SQL las sentencias están formadas por sujeto, verbo y predicado que es lo que más se aproxima a una sentencia en lenguaje natural. Así, *SELECT "*" FROM "tabla" WHERE "condición"* podría ser la traducción sintáctica de *Muéstrame "toda la información" de la tabla "tabla" que tenga la condición "condición"*. Esto facilitará el proceso de interpretación de lenguaje natural y traducción a SQL para el posterior procesamiento de datos.

COMPARATIVA ENTRE SISTEMAS GESTORES DE BASES DE DATOS

Actualmente en las empresas, uno de los principales objetivos es controlar y reducir costes a la vez que intentar mantener un alto nivel de productividad así como reducir la inversión necesaria para adquirir y mantener software de gestión de datos. Por este motivo, las empresas deciden por emplear tecnologías y herramientas que les den solución

a este problema. Teniendo en cuenta las alternativas mercado actuales disponibles, se han diferenciado dos posibles opciones que optimizan el problema del coste-productividad:

Basados en SQL

- Código abierto: MySQL, PostgreSQL, FireBird.
- Comerciales: Oracle, Microsoft SQL Server 2008, Interbase.

Microsoft Access: Sistema gestor de bases de datos diseñado por Microsoft incluido en el paquete Microsoft Office.

En la actualidad, MySQL y Firebird[14] están consolidándose como SGBD más utilizado en los departamentos de IT debido a las ventajas que presentan sobre otros tipos de SGBD Las ventajas principales de estas dos herramientas, frente a otros SGBD basados en SQL como Oracle, PostgreSQL o Interbase, podrían resumirse en las siguientes:

- **Libres**: Son de código abierto y gratuito lo cual empuja a las empresas, sobre todo a las PYMES, a optar por MySQL o Firebird en lugar de Oracle, Interbase y Access.
- **Ligereza**: Consumen muchos menos recursos y son más estables que PostgreSQL.
- **Integridad**: Si se quiere integrar la aplicación en el futuro con una base de datos Web a través de una interfaz en PHP, MySQL y Firebird disponen de más funcionalidad que PostgreSQL.
- **Portabilidad** entre aplicaciones de bases de datos.
- **Multiplataforma**, Access únicamente funciona sobre Windows.
- **Estandarizados**: Son estándares basados en ANSI mientras que Access no lo está.
- **Seguridad**: Access no tiene un sistema de seguridad sólido ni fiable.
- **Portabilidad**: A efectos prácticos, no hay prácticamente diferencias ya que en un SGBD se puede montar un driver OBDC y una DSN pudiendo exportar así tablas de Access a SQL; pero ¿por qué exportar datos si no existe la necesidad de hacerlo?
- **Fiabilidad**: Con Access, cada cliente lee y escribe directamente en las filas de las tablas de datos. Si la máquina de un cliente falla mientras está realizando las

[14] http://www.firebird.sql.org. Último acceso 27-06-2011

operaciones de escritura, probablemente provoque la corrupción de la tabla de datos. Con SQL, el cliente no "habla" directamente con las tablas sino que lo hace con un gestor de datos del servidor que permite que, si una máquina falla, el gestor de datos se dará cuenta de que la operación sobre los datos no ha sido completada y abortará la operación antes de finalizarla, evitando de este modo la pérdida de datos la operación.

- **Integridad de datos**: Esta propiedad se ve mejorada por el uso de triggers.
- Permite definir datos dinámicamente, de forma optimizada.
- Requisitos de Hardware: En términos de hardware, los requisitos de máquina de Access son inferiores a los de SQL. Access presenta problemas cuando trata de manejar grandes cantidades de datos
- Rendimiento superior: Cuando se realiza una petición a un servidor desde Access, se lanza una query que involucrará a ciertas tablas y registros. Esas tablas son descargadas a la máquina del cliente y desde la misma, se filtrarán los resultados de la query. Esto en un proceso de 50.000 registros supone una gran carga para el procesador. En SQL, el filtrado se realiza en el servidor lo cual optimiza el tráfico de datos entre cliente/servidor.
- Escalabilidad**:** Access trabaja bien con hasta 10 clientes pero a partir de ahí, el nivel de rendimiento empieza a caer rápidamente. Con la arquitectura de SQL, miles de clientes pueden tener acceso simultáneamente a la base de datos.

Es por estos motivos por los que, en este proyecto se va a emplear o MySQL o Firebird como alternativa a SQL Server, Interbase, PostgreSQL y MS Access.

FIREBIRD, LA ALTERNATIVA A MYSQL

Firebird es un SGBD que deriva del código fuente de InterBase 6.0 de Borland y, aun no siendo tan conocido como otros sistemas más populares como MySQL o PostgreSQL, a pesar de ser un gestor de base de datos totalmente libre**,** sin licencias duales, exhibe una serie de características que lo hacen muy interesante a la hora de realizar desarrollos de programas empaquetados. Una de las características principales es que esta base puede ser

embebida en el producto final, con lo cual no se le instala un servidor al cliente, cosa que no ocurre con MySQL. Es este aspecto el que marca la distancia entre Firebird y MySQL.

La tecnología de Firebird lleva 20 años funcionando, esto hace que sea un producto muy maduro y estable. El desarrollo de Firebird lleva aparejado la aparición de versiones que incluyen nuevas características y posibilidades. Así se comenzó con la versión 1.0 (simplemente portar el código de internase 6.0 en C), la versión 1.5 (conversión de Firebird a C++), la versión 2.0 (nuevas características como tablas derivadas, etc.), la versión 2.1 (características de gestión de sesiones, etc.) y así hasta llegar a la última prevista, versión 3.0. Estos temas están escritos con las características hasta la versión actual disponible (2.5) por lo que puede que algunas características no estén disponibles en versiones anteriores.

Firebird tiene todas las características y la potencia de un SGBD. Se pueden manejar bases de datos desde unos pocos KB hasta varios Gigabytes con buen rendimiento y casi sin mantenimiento. Sus características principales son:

- Soporte completo de Procedimientos Almacenados y Triggers.
- Las Transacciones son totalmente ACID compliant.
- Integridad referencial.
- Arquitectura Multi Generacional.
- Muy bajo consumo de recursos.
- Completo lenguaje para Procedimientos Almacenados y Triggers (PSQL).
- Soporte para funciones externas (UDFs).
- Poca o ninguna necesidad de DBAs especializados.
- Prácticamente no necesita configuración, ¡sólo instalar y empezar a usarla!.
- Una gran comunidad y muchas páginas donde conseguir buen soporte gratuito
- Opción a usar la versión embebida de un solo fichero, ideal para crear CDROM con catálogos y versiones de evaluación o monousuario de aplicaciones.
- Docenas de herramientas de terceros, incluyendo herramientas visuales de administración, replicación, etc.
- Escritura segura - recuperación rápida sin necesidad de logs de transacciones.

- Muchas formas de acceder a tus bases de datos: nativo/API, driver dbExpress, ODBC, OLEDB, .Net provider, driver JDBC nativo de tipo 4, módulo para Python, PHP, Perl, etc.
- Soporte nativo para los principales sistemas operativos, incluyendo Windows, Linux, Solaris, Mac OS.
- Backups incrementales.
- Disponible para arquitecturas de 64bit.
- Completa implementación de cursores en PSQL.
- Firebird es un SGBD en plataforma cliente/servidor.
- En Windows, se puede ejecutar Firebird como servicio o como aplicación. El instalador puede crear un icono en el panel de control que permite controlar el servidor (iniciarlo, pararlo, etc).

El servidor Firebird viene en tres versiones: SuperServer, Classic y Embedded. Actualmente, Classic es la versión recomendada para máquinas con SMP y algunas otras situaciones específicas.

SuperServer: Comparte su caché para todas las conexiones y usa un hilo de ejecución para cada conexión. Ésta se suele usar en Windows.

Classic: Inicia un proceso de servidor independiente para cada conexión que se haga.

Embedded: Es una interesante variación del servidor. Es un servidor Firebird con todas sus características, empaquetado en unos pocos ficheros. El servidor no necesita instalación.

Obviamente Firebird es muy superior a MySQL e incluso es superior a las versiones express de SQL Server y su licencia comercial permite utilizarlo incluso en programas comerciales sin restricciones GNU Public License (GPL) para que una desarrollador (programando por ejemplo en .NET como en este proyecto) pueda usar Firebird como back-end y usar las librerías de conexión sin tener que liberar su programa bajo la GPL.

FLAMEROBIN. LA INTERFAZ VISUAL DE GESTION PARA FIREBIRD

Para llevar a cabo la gestión de la base de datos mediante Firebird de un modo más sencillo, se hará uso de la herramienta FlameRobin. Esta herramienta se caracteriza porque:

- Es libre.
- Multiplataforma (Linux, Windows, Mac OS X, FreeBSD, Solaris).
- Dependiente únicamente de OpenSources.

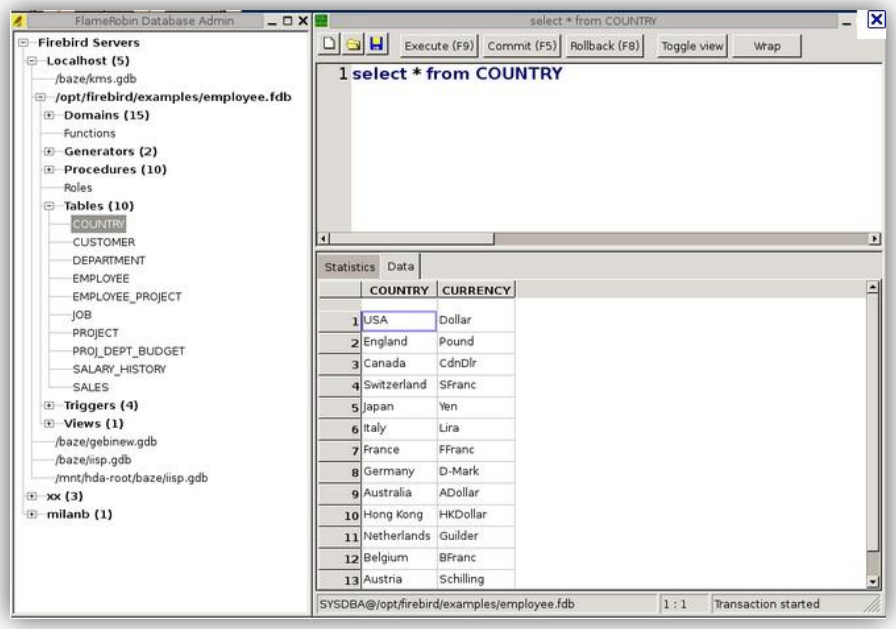


Figura 31. Interfaz FlameRobin.

CONCLUSIONES

Por lo comentado en los apartados anteriores, en este proyecto el sistema gestor de la base de datos que se va a emplear, aunque no se aproveche todo su potencial y funcionalidad, va a ser Firebird. Firebird dispone de multitud de documentación, un manual de referencia bastante amplio (153 páginas), y un portal web donde, entre las opciones brindadas se puede acceder a un foro, descargar los drivers y librerías necesarias para conectar Firebird a la base de datos y a Visual Studio, incluso participar en el desarrollo del proyecto. Esto hace que el proceso de familiarización con la herramienta Firebird sea sencillo y no lleva mucho tiempo.

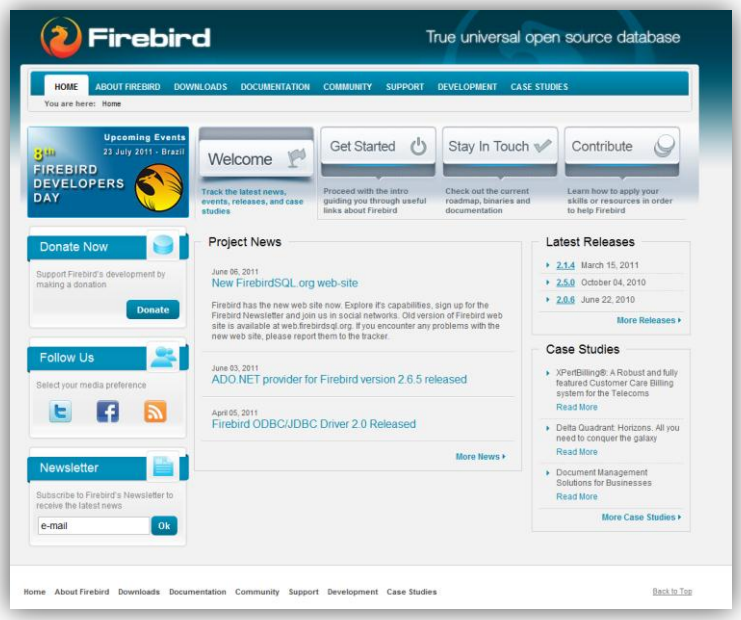


Figura 32. Página web oficial de Firebird http:\\www.firebirdsql.org.

EL MODELADO DE OBJETOS Y COMPONENTES (COM)

Principalmente Modelado de Objetos Componentes implica conectar una aplicación a otra. Todos los componentes de Microsoft Office comparten las herramientas funcionales, permitiendo al usuario realizar cualquier documento e importarlo a otra herramienta como, por ejemplo, la creación de una hoja de cálculo dentro de un documento Word. Esto se debe en gran parte a que las herramientas de MS Office son altamente programables en lenguajes que son mundialmente utilizados tales como Visual Basic, C++ y C#.

La estrategia en el área de herramientas de desarrollo de Microsoft se basa en la oferta de herramientas que permitan integrar las aplicaciones cliente/servidor tradicionales con internet-intranet. Para ello hay una palabra clave que es reutilización, tanto de conocimientos, como de inversión y código; y para ello los componentes son algo básico, entendiendo componentes en un sentido totalmente amplio como aquello que nos permite reutilizar las aplicaciones o parte de ellas.

El desarrollo mediante componentes ofrece ventajas como la independencia del lenguaje, puesto que se pueden desarrollar aplicaciones en diferentes lenguajes o partes de

aplicaciones en distintos lenguajes. También es interesante el poder reutilizar el código existente desarrollado en un lenguaje para crear nuevos componentes que se comunican con los existentes. Otra ventaja es la independencia relativa de la plataforma, en el sentido de que podemos comunicar componentes en una máquina Unix con componentes en Windows.

Una vez que los métodos de aplicaciones remotas sean accesibles a la aplicación original, la aplicación nativa puede compilar y ejecutar los métodos de aplicaciones remotas.

AUTOMATIZACIÓN MEDIANTE CÓDIGO ADMINISTRADO

La automatización de Office desde código administrado es un poco diferente, no es posible directamente enlazar al objeto. La comunicación atraviesa un intermediario "contenedor" que traduce los tipos de datos de .NET nativos y llamadas a tipos COM y las llamadas (y viceversa).

INTEROPERABILIDAD COM EN .NET

Los servicios de interoperabilidad COM de CLR permiten código administrado que se ejecuta en .NET para comunicarse con servidores anteriores de COM (como Office) utilizando un contenedor entre el código administrado y el COM servidor. El contenedor se denomina ensamblado de interoperabilidad. Cuando se agrega una referencia a un servidor COM en un proyecto .NET, Visual Studio .NET utiliza la información para generar código en tiempo de ejecución que el objeto proxy generado en el CLR llamado Envoltura Invocable en Tiempo de Ejecución (del inglés Runtime Callable Wrapper, RCW) importa. Este código administrado se compila en un ensamblado independiente (es decir, el ensamblado de interoperabilidad) y tiene acceso a .NET Framework como si fuese cualquier biblioteca administrada por .NET.

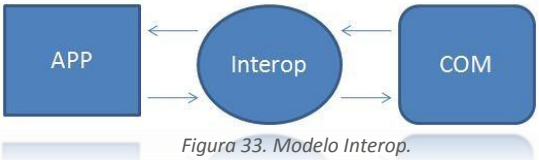


Figura 33. Modelo Interop.

El RCW actúa como elemento intermedio entre el código administrado y el servidor COM.

Aunque Visual Studio .NET puede realizar automáticamente estos ensamblados utilizando el Tlbimp.exe, son ensamblados personalizados y no deben estar registrados en la caché de ensamblados global como objetos globales (http://msdn2.microsoft.com/en-us/library/bd9cdfyx(vs.71).aspx, "Interoperabilidad COM en *Microsoft Developer Network*". Último acceso 28-07-2011).

La ventaja del enlace en tiempo de ejecución es que no es necesario utilizar un RCW, o en un ensamblado de interoperabilidad personalizado de generación y envío, y es independiente de versiones. En problema es la creación de instancias de objeto. Si se utiliza un ensamblado de interoperabilidad, el RCW controla la creación de instancias de objeto cuando se utiliza el operador *New*. En segundo plano, el RCW llama a la rutina apropiada del COM, por ejemplo *CoCreateInstance*, para crear una nueva instancia de esa aplicación para el código que se va a emplear. Si se desea controlar la creación de instancias de objeto explícitamente o enlazar a una instancia de la aplicación ya en ejecución, se puede utilizar *GetObject* en Visual Basic.NET o *GetActiveObject* en Visual C#. Sin embargo, las aplicaciones de Office sólo se registran en la Tabla de Objetos en Ejecución (*del inglés Runtime Object Table, ROT*) si tienen ganado y perdido el foco al menos una vez. Si se está realizando en tiempo de compilación, tiempo y foco van a ser grandes problemas.

EXTENSIBILIDAD DE OFFICE MEDIANTE VISUAL STUDIO .NET

Generar componentes extensibles para ejecutar dentro de la oficina siempre ha sido una parte importante del desarrollo de aplicaciones. Office, Office 2000 y Office XP permiten la extensión de programación de la aplicación host utilizando y presentan capacidades para ampliar el contenido del documento mediante el uso de un reconocedor SmartTag o un componente de acción.

Sin embargo, hay que tener en cuenta lo siguiente:

- El complemento debe registrarse como COM. De forma predeterminada, los DLL administradas que se realizan en Visual Basic o C# no actúan como componentes COM. Para registrarlos, hay que establecer el valor Registrar para interoperabilidad COM en true en la opción "Generar Propiedades de Proyecto".
- Office carga el componente y a continuación lo registra en la Caché de Ensamblados Global (*del inglés Global Assemblies Caché, GAC*). De forma predeterminada, un proyecto de DLL de .NET típico no hace esto.

4.4.3. NET

DEFINICIÓN

En Junio de 2000, la firma de Redmond presentó un entorno independiente del lenguaje, concebido para facilitar el desarrollo de aplicaciones para Windows capaces de trabajar y comunicarse de forma segura y sencilla. Casi una década después, el 70% de los desarrolladores españoles utiliza .NET.

En la actualidad la presencia de este entorno de programación de Microsoft en el mercado de las herramientas de desarrollo de aplicaciones para Windows es abrumadora. Y es que esta plataforma se ha consolidado como una alternativa eficaz a J2EE de Sun Microsystems[15] que, además, proporciona importantes ventajas a la hora de diseñar aplicaciones distribuidas y por ser un framework portable 100% a cualquier plataforma de desarrollo.

[15] http://www,oracle.com/us/sun/

EL ENTORNO DE DESARROLLO .NET

.NET Framework es un componente de Windows para la generación, implantación, y ejecución de aplicaciones en un entorno con un extenso conjunto de lenguajes de programación.

Asimismo, proporciona a los desarrolladores gran parte de la estructura requerida para la generación de software, permitiendo que estos puedan poner un mayor foco sobre el código lógico específico del producto en desarrollo.

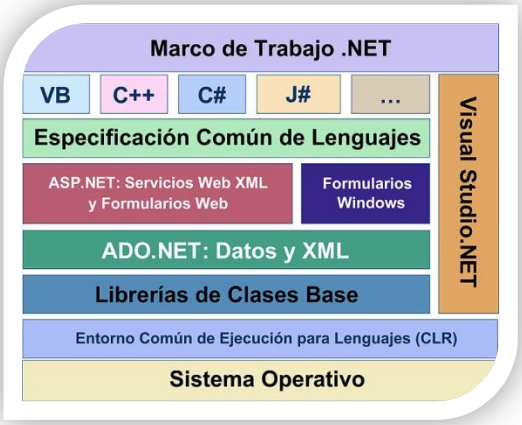


Figura 34. Framework .NET.

A su vez permite la construcción rápida de aplicaciones conectadas que ofrecen experiencias de usuario increíbles, ya que ofrecen bloques de construcción (software pre-fabricado) que resuelven las tareas de programación más frecuentes. Las aplicaciones conectadas construidas sobre los modelos de negocio de .NET Framework procesan de manera efectiva y facilitan la integración de sistemas en entornos heterogéneos.

A continuación se resumen algunas de las ventajas que proporciona .NET:

- **Compila a código intermedio (CIL)** independiente del lenguaje en que haya sido escrita la aplicación e independiente de la máquina donde vaya a ejecutarse.
- **Compilación Just-in-Time (JIT):** El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la

plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.

- Recolección de basura automática.
- Eliminación del uso punteros, en C# no se necesitan.
- No hay que preocuparse por archivos de cabecera ".h".
- No importa el orden en que hayan sido definidas las clases ni las funciones.
- No hay necesidad de declarar funciones y clases antes de definirlas.
- No existen las dependencias circulares.
- Soporta definición de clases dentro de otras.
- No existen funciones, ni variables globales, **todo pertenece a una clase**.
- Todos los valores son inicializados antes de ser usados (automáticamente se inicializan al valor estandarizado, o manualmente se pueden inicializar desde constructores estáticos).
- Concepto formalizado de los métodos *get* y *set*, con lo que se consigue código mucho más legible.
- Gestión de eventos (usando delegados) mucho más limpia.
- El rendimiento es, por lo general, mucho mejor.
- CIL (el lenguaje intermedio de .NET) está **estandarizado**, mientras que los bytecodes de Java no lo están.
- Compilación condicional.

Es cierto que no todo son ventajas, ya que procesos como la recolección de basura de .NET o la administración de código, introducen factores de sobrecarga que repercuten en la demanda de más requisitos del sistema. Como se ha visto, el código administrado proporciona una mayor velocidad de desarrollo y mayor seguridad de que el código sea bueno. En contrapartida, el consumo de recursos durante la ejecución es mucho mayor, aunque con los procesadores actuales esto cada vez es menos inconveniente. Además, el nivel de administración del código dependerá en gran medida del lenguaje que utilicemos para programar. Por ejemplo, mientras que Visual Basic .Net es un lenguaje totalmente administrado, C-Sharp permite la administración de código de forma manual, siendo por defecto también un lenguaje administrado, mientras que C++ es un lenguaje no

administrado en el que se tiene un control mucho mayor del uso de la memoria que hace la aplicación.

Particularizándolo a nuestro proyecto, ha sido utilizado el entorno .NET Framework 3.5, ya que construye más que su predecesor .NET Framework 3.0 y del cual existe una extensísima documentación en Microsoft Developer Network [16](MSDN). Las mejoras están basadas en áreas fundamentales como la biblioteca básica de clases, Windows Workflow Foundation (WWF), Windows Communication Foundation (WCF), Windows CardSpace y Windows Presentation Foundation (WPF). Con respecto a éste último, WPF es una de las novedosas tecnologías de Microsoft y uno de los pilares de Windows Vista, aunque su uso no es exclusivo de este sistema operativo ya que puede ser utilizado también sobre Windows XP, Seven. WPF es toda una plataforma que da soporte para poder colocar en sus aplicaciones una amplia riqueza visual e interactiva, e integrarlas fácilmente con la lógica del negocio.

ARQUITECTURA DE APLICACIONES .NET

.NET Framework consta de cuatro partes principales (*MSDN 2008*):

COMMON LANGUAGE INFRASTRUCTURE (CLI)

El código puede ser escrito en cualquier lenguaje compatible con .NET ya que siempre se compila en código intermedio (MSIL).

COMMON LANGUAGE RUNTIME (CLR)

El CLR se encuentra en el corazón del entorno, de hecho es una máquina virtual encargada de proporcionar una capa de abstracción entre el bytecode de .NET y la plataforma hardware sobre la que se ejecuta desempeñando una función tanto en tiempo de ejecución como en el proceso de desarrollo de componentes.

[16] http://www.msdn.microsoft.com/. Último acceso 28-06-2011

En tiempo de ejecución, el CLR es responsable de la administración de memoria, iniciar y detener procesos y subprocesos, cubrir las dependencias entre estos, etc.

Durante la etapa de desarrollo, el CLR cobra relevancia simplificando el trabajo del desarrollador al contribuir con características del tipo de la administración del ciclo de vida, la nomenclatura de tipos, y la gestión de excepciones entre distintos tipos de lenguajes. Esto permite reducir la cantidad de código que debe escribir un desarrollador para convertir sus productos en componentes reutilizables.

BIBLIOTECA DE CLASES BASE

La biblioteca de clases base de .NET presenta un conjunto unificado, orientado a objetos, jerárquico y extensible a bibliotecas de clases (API). El entorno de trabajo permite unificar modelos como Microsoft Foundation Classes y las Windows Foundation Classes para permitir a programadores de diversos lenguajes poder tener acceso a las bibliotecas de clases. La creación de un conjunto de API comunes evita limitaciones en la utilización de los lenguajes de programación en el entorno de trabajo, dejando al desarrollador la elección del lenguaje que desee utilizar sin ningún tipo de restricción.

Figura 35. Librería de clases .NET.

ASP .NET (ACTIVE SERVER PAGES)

ASP.NET es un framework desarrollado y comercializado por Microsoft que sucedió a ASP, que está cimentado sobre el CLR y que permite a los desarrolladores

construir sitios web dinámicos, aplicaciones web y servicios web XML. Entre las principales ventajas de ASP.NET se encuentran:

- **Rendimiento**: La aplicación compila en una sola vez al lenguaje nativo y luego, en cada petición, tiene una compilación Just In Time, es decir se compila desde el código nativo, lo que permite mucho mejor rendimiento.
- **Rapidez en programación:** Mediante diversos controles podemos, con unas pocas líneas y en menos de 5 minutos, mostrar toda una base de datos y hacer rutinas complejas.
- **Servicios Web:** Proporciona herramientas para compartir datos e información entre distintos sitios.

RECOLECTOR DE BASURA (GARBAGE COLLECTOR)

Este recolector de basura es el mecanismo que se encarga de reasignar la memoria de objetos que ya no estén siendo referenciados. Los beneficios de la recolección de basura son indiscutibles, aumento de la fiabilidad, la disociación de la gestión de memoria de diseño de la interfaz de clase, y menos tiempo de desarrollo persiguiendo errores de gestión de memoria. Este proceso no tiene que conocerlo ni realizarlo manualmente el programador ya que .NET sabe cuándo hacerlo.

4.4.4. VISUAL STUDIO TOOLS FOR OFFICE (VSTO)

DESCRIPCIÓN

Hoy en día es posible usar las herramientas para desarrolladores de Microsoft Office en Visual Studio 2010 para crear aplicaciones de .NET Framework que extiendan Microsoft Office 2010 y 2007 Microsoft Office System. Estas aplicaciones también se denominan soluciones de Office.

Las herramientas para desarrolladores de Office proporcionan características que le ayudan a crear soluciones de Office para satisfacer una variedad de necesidades empresariales. Las herramientas incluyen plantillas de proyecto para ayudarle a crear

soluciones de Office con Visual Basic o Visual C#, y diseñadores visuales que le ayudan a crear interfaces de usuario personalizadas para las soluciones de Office.

ARQUITECTURA VSTO

Para poder ejecutar las personalizaciones de nivel de documento creadas con las herramientas de desarrollo de Office en Visual Studio, los equipos de los usuarios finales deben tener instalado el motor en tiempo de ejecución de Microsoft Visual Studio Tools para Office. El Motor en tiempo de ejecución de Microsoft Visual Studio Tools para Office incluye componentes no administrados que cargan el ensamblado de personalización además de un conjunto de ensamblados administrados.

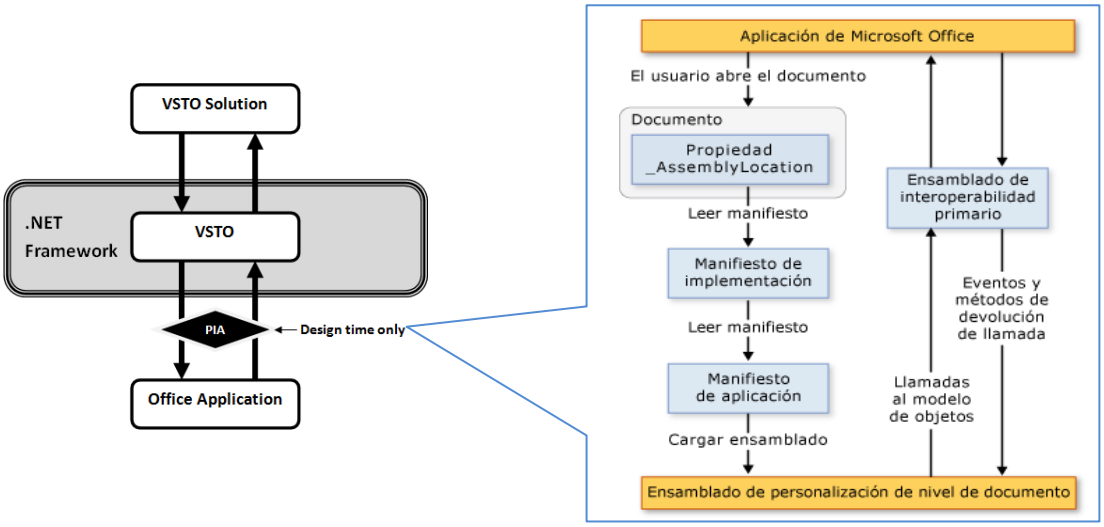


Figura 36. Arquitectura VSTO.

Estos ensamblados administrados o PIA proporcionan el modelo de objetos COM que utiliza el código de la personalización para automatizar y extender la aplicación host, se agregan automáticamente al proyecto una referencia a ellos y son necesarios para compilar el proyecto.

4.5. ARQUITECTURA

En este apartado se pretende plasmar cuál será la arquitectura de la aplicación a nivel lógico de la base de datos, de la aplicación propiamente dicha, cómo se van a comunicar las capas del MVC, qué esquema estructural van a tener los elementos empleados y por último se va a detallar el flujo de información entre ellas.

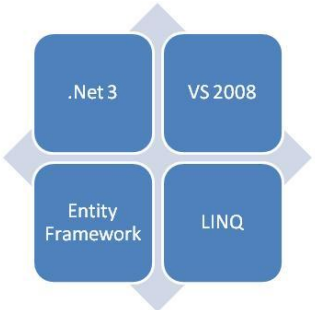


Figura 37. Modelo arquitectura NQPL.

4.5.1. ENTRELAZADO LÓGICO ENTRE LA BASE DE DATOS FIREBIRD Y VISUAL STUDIO

Para que Visual Studio 2010 pueda comunicarse con una base de datos Firebird, el equipo de desarrollo ha implementado una biblioteca de clases que, una vez cargadas y referenciadas en un proyecto de Visual Studio, habilitan la creación de objetos Firebird y consecuentemente la comunicación e interacción entre la base de datos y el entorno de desarrollo.

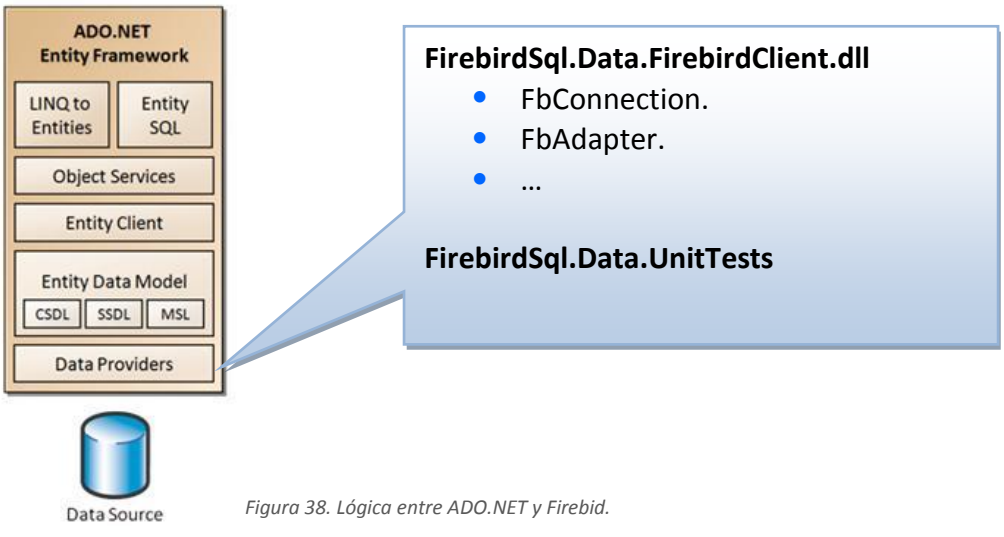


Figura 38. Lógica entre ADO.NET y Firebid.

FBConnection: Objeto que permite establecer la conexión con la base de datos de Firebird. Ejemplo de conexión:

```
case "FDB":
        string sConnectionString =
        "User=SYSDBA;" +
        "Password=masterkey;" +
        "Database=" + sBrowserPath + ";" +
        "DataSource=localhost;" +
        "Port=3050;" +
        "Dialect=3;" +
        "Charset=NONE;" +
        "Role=;" +
        "Connection lifetime=15;" +
        "Pooling=true;" +
        "MinPoolSize=0;" +
        "MaxPoolSize=50;" +
        "Packet Size=8192;" +
        "ServerType=0";

        // Creamos una conector de Firebird para acceder a la base de datos.
        FbConnection oFBConexion = new FbConnection(sConnectionString);
```

FBAdapter: Objeto que permite lanzar instrucciones desde Visual Studio a la base de datos con la que se ha conectado previamente.

```
FbDataAdapter oDataAdapter = new FbDataAdapter("SELECT * FROM " +
SelectedTable, oFBConexion);
```

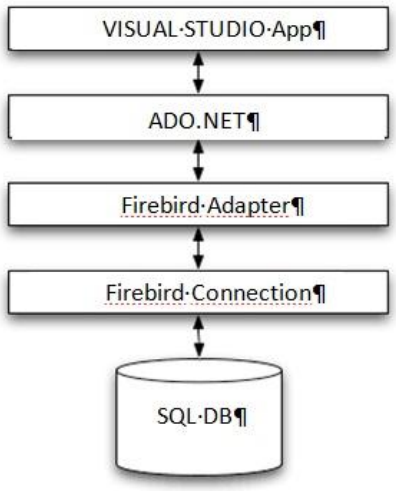


Figura 39. Firebird data provider, Ado.Net and .NET.

La información contenida en el FBAdapter, entre otras, contiene los datos resultantes de la operación realizada. Estos datos se pueden volcar en un objeto ADO.NET para su posterior manipulación.

Figura 40. Flujo datos FBAdapter y ADO.NET.

Una vez obtenido el objeto ADO.NET se pueden realizar operaciones LINQ sobre dichos datos y obtener los resultados deseados, como por ejemplo,

```
IEnumerable<string> oHasCondition;
            oHasCondition = from x in oOutput
                     where (x == "X") || (x == "Y") || (x == "J")
                     select x;
```

4.5.2. VISUAL STUDIO .NET Y ADO.NET

El IDE de Visual Studio .NET puede hacer uso de los objetos proporcionados por la biblioteca de clases de ADO.NET, más en concreto con la parte de DataSets. La manipulación de los datos de los objetos es plena desde Visual Studio y permite, debido a su arquitectura MVC, bindar estos objetos a la capa de XAML de la aplicación y visualizar los resultados.

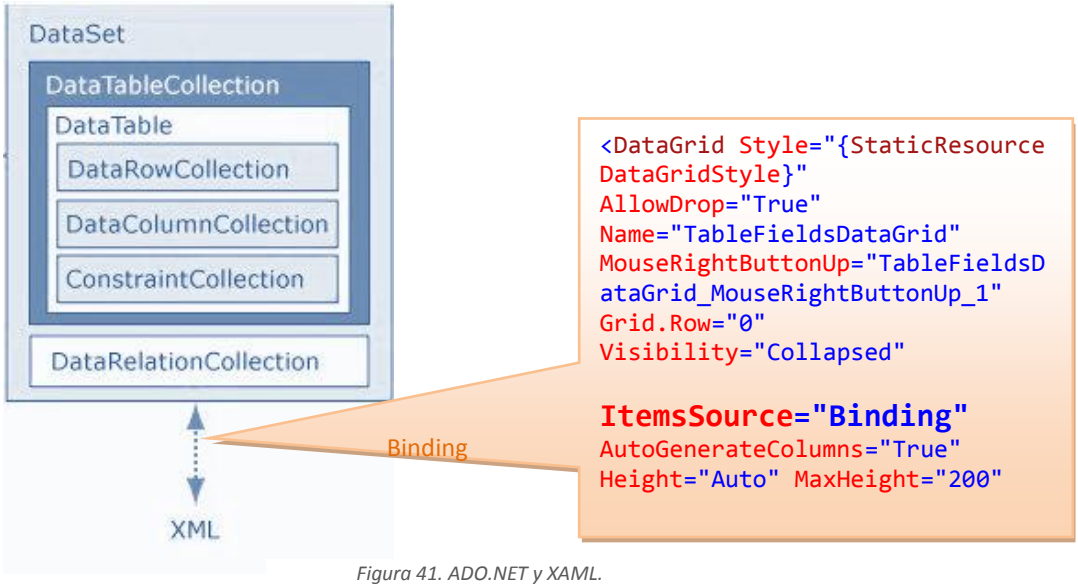


Figura 41. ADO.NET y XAML.

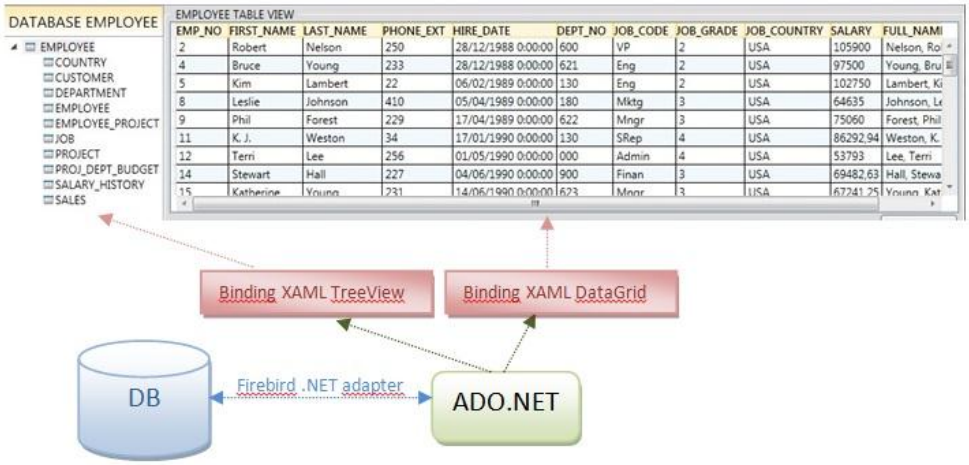


Figura 42. Arquitectura Firebird, ADO.NET y XAML.

4.5.3. VISUAL STUDIO Y MICROSOFT OFFICE

La conexión entre un componente Microsoft Office y Visual Studio se realiza mediante un ensamblado de interoperabilidad intermedio llamado. (Microsoft.Office.Interop:Excel, Microsoft.Office.Interop.Word, etc). Esta biblioteca permite crear y modificar objetos COM desde Visual Studio en un contenedor intermedio (*ver INTEROPERABILIDAD COM EN .NET y Figura 36. Arquitectura VSTO.*).

Este componente permitirá que se incrusten objetos COM en soluciones Office en forma de Add-ins y que se obtenga una plena libertad de comunicación e interacción entre dichos Add-ins y los componentes de Microsoft Office.

4.5.4. ARQUITECTURA GLOBAL

Como resultado de la combinación de los esquemas y concentos lógicos de entrelazados descritos anteriormente, se presentará a continuación una figura que dejará claramente definida cuál será la arquitectura global del sistema. Además, en la figura se definirán los flujos de datos entre los distintos componentes

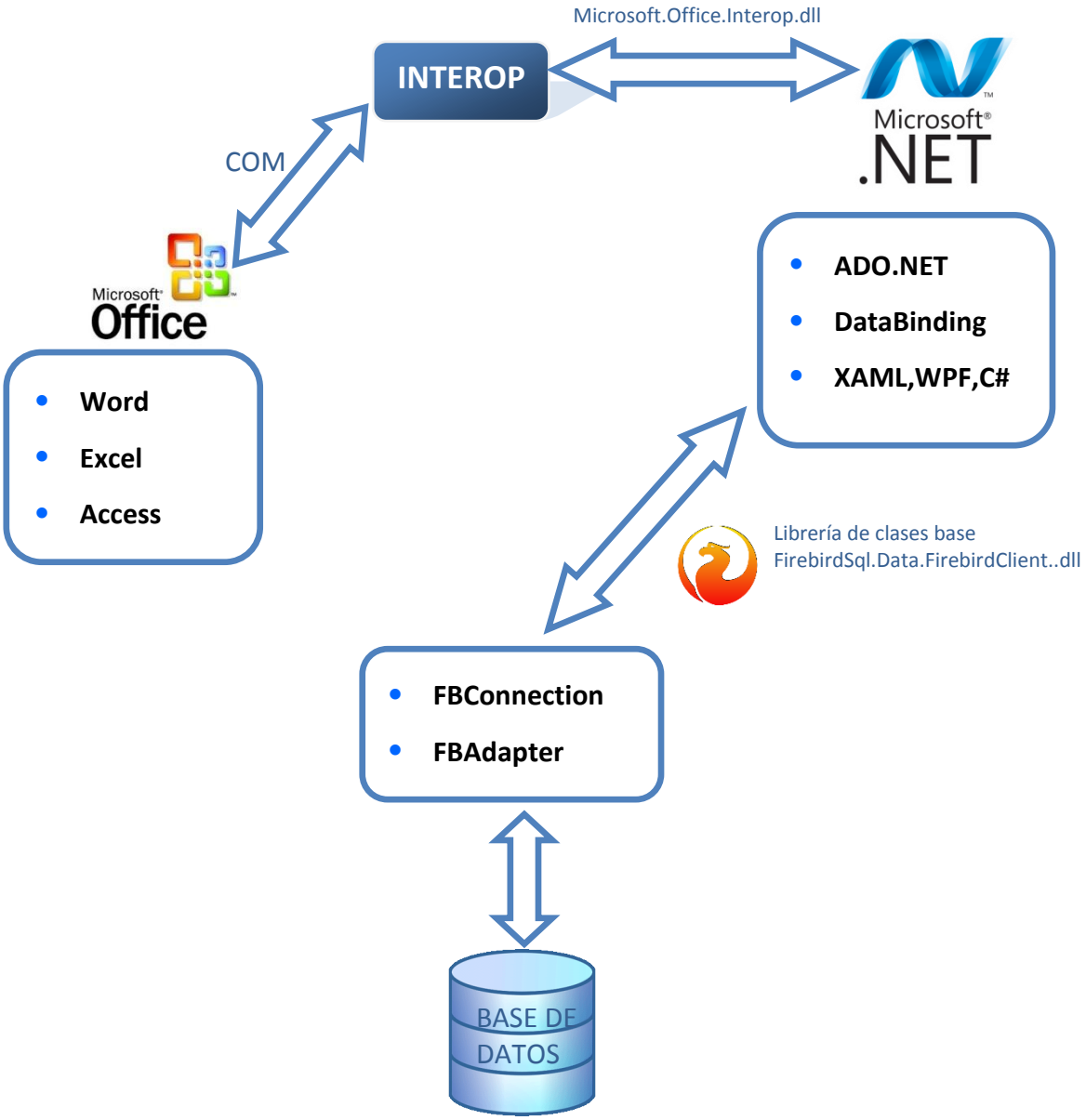


Figura 43. Arquitectura general del sistema.

CAPÍTULO V

CONCLUSIONES Y LÍNEAS FUTURAS

5.1. CONCLUSIONES

En este apartado se van a exponer las conclusiones obtenidas tras la finalización del proyecto desarrollado. Se abordarán una serie de conclusiones generales acerca del proyecto y posteriormente las conclusiones obtenidas a raíz de los problemas identificados.

5.1.1. CONCLUSIONES GENERALES

La relevancia del trabajo desarrollado radica en la necesidad de obtener una aplicación que facilite la traducción del lenguaje natural a un lenguaje declarativo de manipulación de datos con el fin de aprovechar la nueva funcionalidad. El objetivo de la creación de una aplicación que posea un módulo traductor de lenguaje natural a sentencias SQL es la de permitir, en principio, realizar consultas y manipular información sobre una base de datos SQL teniendo en cuenta las posibles limitaciones de conocimientos técnicos del usuario.

Todo proceso de ingeniería de software consta de varios ciclos o fases. Para un proyecto de dimensiones intermedias como es el caso, resulta beneficioso descomponer dichas fases de desarrollo en ciclos, actividades y tareas que contribuyan a facilitar el trabajo, mejoren la eficiencia y permitan mantener un seguimiento de la situación en la que se encuentra el proyecto en cada momento.

Durante el primer ciclo de este Proyecto de Fin de Carrera se desarrolló un trabajo de investigación y estudio de qué se quería hacer en él, que alternativas similares habría al proyecto y cómo se iba a abordar. Durante el segundo ciclo se decidió que marco de trabajo se iba a utilizar de tal modo que facilitase el proceso de implementación. Finalmente, en el tercer ciclo el objetivo principal consistirá en la implementación de la solución del software, objetivo principal de este proyecto.

Gracias a disponer de un framework adaptado y adaptable el trabajo realizado se ha podido realizar con mayor eficiencia y rapidez.

En cuanto a la planificación temporal y empleo de recursos para un proyecto de ingeniería de software, tanto materiales como personales, es una tarea inicial crítica para que el posterior desarrollo de la solución sea el adecuado. Este primer proceso requiere un análisis exhaustivo acerca del sistema a implementar para conseguir recopilar la información suficiente y necesaria para poder abordar decisiones fundamentales y que decidirán el rumbo del mismo. Esto implica además del estudio de qué personal técnico, externo o interno a la empresa, es el más adecuado y se ajusta más al proyecto, con el fin de construir un entorno de trabajo que mejor se acople a estas necesidades.

Una de las conclusiones más importantes que se han obtenido en el proceso de desarrollo de este software es que hay que tener en cuenta modelos previos y similares implementados para orientarse, no qué vamos a hacer, sino cómo se podría hacer. Por ejemplo, durante el desarrollo de un proyecto de reingeniería de software en el que he participado se implementó un editor de sentencias para interactuar con una base de datos, bajo el mismo entorno de desarrollo que el empleado en este proyecto. Esto resulta útil a la hora de tomar como referencia el modelo conceptual de análisis de sentencias y la estructura visual de la aplicación previamente diseñados. Con esto no se quiere decir que se vayan a replicar proyectos anteriores, sino que se van a tener en cuenta para encaminarse hacia una solución exitosa.

Como última consideración, es de vital importancia que quede constancia de todo lo que se ha hecho en el proyecto para que futuras mejoras o desarrollos sobre el mismo puedan realizarse de una manera eficaz y exitosa. Para ello se recomienda encarecidamente que exista una documentación lo más detallada posible acerca de lo realizado desde el comienzo del proyecto hasta el final. Esto facilitará a personal que se incorpore a posteriori instruirse lo suficiente como para saber sobre qué están trabajando, herramientas utilizadas y aspectos básicos de la aplicación evitándose así que el proceso de formación sea demasiado tedioso y largo.

5.1.2. CONCLUSIONES EXTRAÍDAS DE LOS PROBLEMAS IDENTIFICADOS

A lo largo del Capítulo I del presente Proyecto Fin de Carrera, se han enumerado una sucesión de problemas que afrontar para el desarrollo de la solución propuesta. A continuación se listan las conclusiones y soluciones obtenidas para cada uno de estos problemas:

- Problema con el usuario inexperto: La solución a este problema es cuestión de inversión de tiempo, una buena formación y la correcta documentación de software.
- Problema con la ineficacia, poca fidelidad de traducción de LN a SQL: empleo de algoritmos complejos, estructuras de grafos y transiciones de lenguaje mediante el uso de autómatas (análisis y descomposición semántica, sintáctica y gramatical). Este problema siempre estará presente ya que la lingüística varía muy poco a lo largo de los años. EL problema será cada vez menor cuanto más complejos y elaborados sean los mecanismos de traducción.
- Problema de compilación de la aplicación multiplataforma: Con la salida al mercado de Mono, este problema de portabilidad ha dejado de existir ya que mono es un entorno multiplataforma que permite ejecución de aplicaciones .NET.

5.2. LÍNEAS FUTURAS

Un trabajo como el desempeñado en este proyecto constituye una fuente de posibles líneas de trabajo en el futuro, tanto en la mejora del módulo de comprensión del lenguaje natural como en la incorporación de un módulo de decodificación acústica, sin olvidar nuevas posibilidades que conduzcan al desarrollo de un "Sistema de Diálogo Hablado" que permita implementar aplicaciones en las que se produzca una verdadera interacción hombre-máquina. Todos estos posibles proyectos futuros están orientados a dar solución a problemas de ambigüedad lingüística e incorporar mecanismos más complejos de traducción que permitan combatir errores originados por la interacción dialogada del usuario con el sistema.

Realmente son muchas las posibles líneas futuras que se nos ocurre proponer como resultado de la experiencia acumulada en el desarrollo del presente trabajo. Sin embargo, vamos a destacar sólo aquellos aspectos que nos parecen más interesantes con el objetivo fundamental de alcanzar una base de conocimiento y experiencia lo suficientemente rica como para habilitar el futuro desarrollo de sistemas de diálogo hombre-máquina basados en lenguaje natural dentro de dominios semánticos más restringidos:

- Mejora de la eficiencia y ahorro en memoria en el módulo de traducción gramatical.
- Mejora del modelado lingüístico permitiendo traducir gramáticas con mayor cobertura.
- Permitir la interacción entre los módulos que conforman el sistema de comprensión, favoreciendo el tratamiento de las ambigüedades y de los errores en el sistema.
- Incorporar conocimiento del contexto del discurso y conectar los distintos módulos con un módulo de control global y manejo del diálogo, que permita interaccionar con el usuario cuando el sistema es incapaz de comprender completamente una consulta o resolver la traducción. Esto es que el módulo se comunique mediante el módulo acústico con el usuario y viceversa.
- Implementación de un sistema interlingua que incluya metodologías de descomposición gramatical con árboles de derivación y analizadores de gramáticas formales (autómatas y grafos).

- Integración de un sistema basado en sintáxis o en gramáticas semánticas.
- Desarrollar nuevas aplicaciones en otros dominios semánticos restringidos que validen la arquitectura propuesta, resolviendo aquellas deficiencias que limiten en exceso su generalidad o flexibilidad, mejorando las primitivas de los lenguajes de reglas, mejorando los intérpretes de reglas, desarrollando modelos que permitan la incorporación de restricciones semánticas y del dominio de aplicación que permitan la detección de problemas planteables al usuario a través del módulo de diálogo.
- Codificación de un sistema de captura visual del lenguaje de sordos para codificar el lenguaje gesticular de personas con discapacidad auditiva y del habla y así permitir la interacción sin restricciones con el módulo de traducción y la interfaz.
- Implementar un módulo que permita transacciones de datos de una tabla Access a SQL.
- Compatibilizar la aplicación con Mono[17] para posibilitar la ejecución de la aplicación en multiplataforma (Linux, Mac OS). Aquí podría surgir un problema de integración del Add-in con Microsoft Office. La solución prevista pasaría por crear un Add-in que fuera compatible con LibreOffice.
- Submódulo de interfaz de Web. Sobrepasar los límites de una base de datos local y poder extender la usabilidad de la aplicación a toda la red.
- Integración y soporte multilenguaje. Añadir más lenguajes de traducción, no únicamente castellano e inglés.
- Integrar un sistema de reconocimiento de voz que posibilite el proceso de análisis gramatical, partiendo de un LN hablado, y que proporcione un resultado acústico a la vez que escrito para habilitar una conversación, propiamente dicho, entre máquina y usuario.

[17] http://www.mono-project.com/

DEFINICIONES

ActiveX Entorno para definir componentes de software.

Asynchronous JavaScript and XML Aplicaciones que se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Ambigüedad Que puede entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión.

Anáfora Tipo de deixis que desempeñan ciertas palabras para recoger el significado de una parte del discurso ya emitida; p. ej., lo en: “dijo que había estado, pero no me lo creí”.

AWK Lenguaje de programación de procesamiento de textos.

Back-end Concepto que hace referencia a la ejecución de la entrada de un proceso y genera la salida del mismo. Suele estar oculto al usuario final.

Biblioteca de enlace dinámico Archivos con código ejecutable que se cargan bajo demanda de un programa o sistema operativo.

Binario Sistema de numeración en el que los dígitos se representan utilizando únicamente el 0 y el 1.

Button Botón.

Bytecode Código intermedio más abstracto que el código máquina.

Clase Abstracción de un objeto.

Code behind Modelo de diseño que mantiene el fichero de interfaz gráfica y el del código con la funcionalidad separados.

Coding Guidelines Documentación que recoge un convenio de codificación para evitar diferencias en el código entre distintos programadores del mismo equipo. El objetivo es unificar estilos de programación para conseguir un código homogéneo.

DataSet Clase que representa una caché de memoria interna de datos.

Debugger Depurador. Programa usado para probar y eliminar errores de otros programas examinando código.

DirectX Colección de APIs desarrolladas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

Drag 'n drop Arrastrar y soltar.

Driver Controlador de dispositivo que permite al ordenador interactuar con un periférico.

C Es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix.

Caché de ensamblados global Almacena los ensamblados designados específicamente para ser compartidos por varias aplicaciones del equipo.

Check-out Se crea una copia de trabajo local desde el repositorio.

Check-in Sucede cuando una copia de los cambios hechos a una copia local es escrita o integrada sobre repositorio.

Código máquina Código directamente interpretable por un circuito micro programable**,** como el microprocesador de una computadora.

Código fuente Conjunto de líneas de texto de un programa que son las instrucciones que debe seguir la computadora para ejecutar dicho programa.

Código intermedio Código que generan algunos compiladores de un análisis sintáctico y semántico, algunos y que es una representación intermedia explícita del programa fuente.

Complemento de aplicación Aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

Componente integrado Elemento integrado en un paquete de elementos con los que trabaja en conjunto.

Computadora Máquina electrónica que recibe y procesa datos.

Consulta formal Tipo de lenguaje que un usuario realiza para obtener información de una base de datos.

C++ Lenguaje híbrido de programación que extiende C a mecanismos que puedan manejar objetos.

Deixis Que se produce mediante anáfora.

Elipsis Supresión de algún término.

Framework O marco, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

GNU Public License Licencia orientada principalmente a proteger la libre distribución, modificación y uso de software.

Hardware Conjunto de partes tangibles de una computadora.

Hipertexto Texto que en la pantalla de un dispositivo electrónico conduce a otro texto relacionado.

HTML dinámico Del inglés Dynamic HTML o DHTML, es un conjunto de técnicas que permiten unificar en el desarrollo web Javascript, Html, objetos DOM y hojas de estilo en cascada (del inglés Cascade Style Sheet, CSS).

Ingeniería de Software Disciplina o área de la Ingeniería que ofrece métodos y técnicas para desarrollar y mantener software.

Interfaz Conexión entre dos ordenadores o máquinas de cualquier tipo dando una comunicación entre distintos niveles. Interfaz también hace referencia al conjunto de métodos para lograr interactividad entre un usuario y una computadora.

Internet Conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP.

Internet Information Services Servidor web y conjunto de servicios para el sistema operativo Windows.

Java Lenguaje de programación de alto nivel, orientado a objetos.

JavaBean Modelo de componentes creado por Sun Microsystems.

Javascript Lenguaje de programación ejecutado por un intérprete, no por un compilador.

Just-in-Time Consiste en traducir el bytecode a código máquina nativo en tiempo de ejecución.

Lenguaje ensamblador Del inglés assembly language o assembler. Lenguaje de programación de bajo nivel para las computadoras, microprocesadores y otros circuitos integrados programables.

Lenguaje computacional Lenguaje que utiliza la informática para estudiar y tratar el lenguaje humano.

Lenguaje de alto nivel Lenguaje que expresa algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de a la capacidad ejecutora de las máquinas**.**

Lenguaje de consulta integrado Proyecto de Microsoft que agrega consultas nativas semejantes a las de SQL a los lenguajes de la plataforma .NET, inicialmente a los lenguajes Visual Basic .NET y C#.

Lenguaje de programación Idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras.

Lenguaje formal Lenguaje técnico y específico que el hombre ha desarrollado para expresar las situaciones que se dan en cada área del conocimiento científico.

LISP Lenguaje de programación favorito en la investigación de la Inteligencia Artificial.

Lenguaje natural Medio que se utiliza de manera cotidiana para establecer nuestra comunicación con las demás personas.

Léxico Lista de palabras; las palabras utilizadas en una región específica, las palabras de un idioma, o incluso de un lenguaje de programación.

Linux Linux es un núcleo de sistema operativo libre tipo Unix.

MAC Abreviatura de Macintosh. Nombre con el cual nos referimos a cualquier computadora creada por la compañía Apple Inc.

Mainframe Ordenador de gran tamaño. En términos de bases de datos es un servidor.

Microsoft Empresa multinacional de origen estadounidense dedicada al sector de la informática.

Microsoft Foundation Classes conjunto de clases que provee un acceso más sencillo a las API de Windows.

Mono Proyecto de código abierto que tiene como objetivo crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET.

Windows Foundation Classes Marco de desarrollo orientado a objetos que encapsula, simplifica y unifica los modelos de programación Win32 y HTML dinámico.

Microsoft Office Suite de oficina que abarca e interrelaciona aplicaciones de escritorio, servidores y servicios para los sistemas operativos Microsoft Windows y Mac OS X.

Microsoft Office System Paquete de productividad de Microsoft que Incluye programas, servidores, servicios y soluciones diseñados para trabajar conjuntamente (InfoPath, Publisher, Word, Access,One Note etc.).

Microsoft Windows Serie de sistemas operativos desarrollados por Microsoft.

Modelo E/R Herramienta para el modelado de datos de un sistema de información.

Modelo de datos lógico Orientado a operaciones. Modelo E/R.

Modelo de datos físico Estructuras de datos de bajo nivel.

Multiproceso Simétrico Arquitectura de computadoras en las que dos o más procesadores comparten la memoria central.

Objeto Instancia de una clase que realiza tareas.

OpenSource Término que hace referencia a productos y software finales cuyo código fuente es compartido.

Página web Documento o fuente de información, generalmente en formato HTML y que puede contener hiperenlaces a otras páginas web. Dicha página web, podrá ser accesible desde un dispositivo físico, una intranet, o Internet.

Paquete de productividad Colección de aplicaciones diseñadas para ahorrar tiempo a los usuarios en la oficina, en el colegio y en casa.

Paradigma Cada uno de los esquemas formales en que se organizan las palabras nominales y verbales para sus respectivas flexiones.

Parser Componente del compilador o el intérprete que analiza estructuras gramaticales y las valida construyendo una estructura de datos con tokens.

Parsing Proceso de análisis de un texto de una gramática formal para obtener su estructura gramatical.

Perl Lenguaje de programación basado en C, AWK y Lisp.

Plugin Complemento de aplicación.

Pragmática Subcampo de la lingüística que estudia el modo en que el contexto influye en la interpretación de un significado.

Propiedad tipada Propiedad con un tipo de datos definido y restringido.

Protocolo Conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.

Redundancia Cierta repetición de la información contenida en un mensaje, que permite, a pesar de la pérdida de una parte de este, reconstruir su contenido.

Regla Mnemotécnica Oración corta y fácil de recordar que ayuda de manera artificiosa a relacionar palabras, con el objetivo de memorizar conceptos con más facilidad.

Repositorio Depósito o archivo centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

Ribbon O cinta, es una barra de comandos que organiza las características de una aplicación en una serie de pestañas en la parte superior de la ventana de aplicación.

Semántico Significado o interpretación del significado de un determinado elemento, símbolo, palabra, expresión o representación formal.

Shell Interfaz empleada para interactuar con el núcleo del sistema operativo.

Silverlight Estructura para aplicaciones web desarrollada por Microsoft que agrega nuevas funciones multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones e interactividad.

Sistema Operativo Programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones.

Smart client Término que hace referencia a un entorno de aplicación que no requiere instalación, que se actualiza automáticamente y que tiene una interfaz de escritorio.

SmartTag Etiqueta inteligente de Microsoft Office Word.

Software Soporte lógico de una computadora.

Standalone Aplicado al software, dícese de aquellas aplicaciones que no tienen dependencias, ni de red ni con otro software.

TCP/IP Estándar de comunicaciones entre redes.

Token Instancia de un tipo usada para la distinción de símbolos en lenguajes formales.

Trigger O disparador, es un procedimiento que se ejecuta cuando se cumple una condición establecida al ejecutar una operación.

Tupla Secuencia ordenada de objetos en una lista.

Solaris Sistema operativo creado por Sun Microsystems.

Unix Sistema operativo portable, multitarea y multiusuario.

Visual Basic Lenguaje de programación dirigido por eventos.

Web Malla, Red.

Win32 API de Windows de 32 bits.

Windows CardSpace Tecnología que sale junto con el .NET Framework 3.0, provee a los usuarios la habilidad de manejar sus identidades digitales.

Windows Communication Foundation Es una API del framework de .NET para aplicaciones orientadas a servicios.

Windows Forms Interfaz Gráfica de Aplicación (API) que está incluida como parte del marco de desarrollo .NET de Microsoft.

Windows Workflow Foundation Tecnología de Microsoft que p proporciona una API y un diseñador para implementar procesos de ejecución largos como flujos de trabajo.

Windows Presentation Foundation Tecnología de Microsoft que permite el desarrollo de interfaces de interacción en Windows tomando las mejores características de las aplicaciones Windows y de las aplicaciones web.

.doc Extensión de archivo nativo, cerrado y muy utilizado por Microsoft.

.fbd Extensión de las bases de datos de Firebird.

.ico Extenxión archivos de imagen de icono.

.NET Framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

.pdf Formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.

ACRÓNIMOS

AJAX Asynchronous JavaScript And XML (JavaScript y XML Asíncrono).

ANSI American National Stándards Institute (Instituto Nacional Estadounidense de Estándares).

API Application Programming Interface (Interfaz de Programación de Aplicaciones).

BD Base de Datos (*del inglés DataBase, BD o BBDD*).

BIT Binary digIT (Dígito binario).

CEDINET Centro Nacional de Investigación y Desarrollo Tecnológico.

CPL Combined Programming Language (Lenguaje de Programación Combinado).

DBMS DataBase Management System (Sistema Gestor de Base de Datos o SGBD).

DLL Dynamic-Link Library (Biblioteca de Enlace Dinámico).

DSN Data Source Name (Nombre de Origen de Datos).

GNU GNU is Not Unix.

GUI Graphical User Interface (Interfaz Gráfica de Usuario).

HTML Hypertext Markup Language (Lenguaje de Marcado de Hipertexto).

AI Artificial Intelligence (Inteligencia Artificial).

IDE O Entorno de Desarrollo Integrado es un programa informático compuesto por un conjunto de herramientas de programación que ha sido empaquetado como aplicación.

ILNBD Interface de Lenguaje Natural a Base de Datos.

INTELLECT INTErmediate-Language LEvel C Translator (Traductor C de Nivel de Lenguaje Intermedio).

IIS Internet Information Services (Servicios de Información de Internet).

J2EE Java 2 Enterprise Edition.

LINQ Language INtegrated Query (Lenguaje de Consulta Integrado).

LISP LISt Processing (Procesamiento de Listas).

LQL Logical Query Language (Lenguaje de Consulta Lógico).

LIL Logic Interface Language (Interfaz Lógico del Lenguaje).

LN Lenguaje Natural.

MS Microsoft.

MSIL Microsoft Intermediate Language (Lenguaje Intermedio de Microsoft o código intermedio de Microsoft).

OLEDB Object Linking and Embedding for Databases (Enlace e incrustación de objetos para bases de datos).

OQL Object Query Language (Lenguage de Consulta de Objetos).

OS Del inglés Operating Sistem (Sistema Operativo).

PDF Portable Document Format (Formato de Documento Portátil).

PHP Personal Home Page Tools (Herramientas Personales de Página de Inicio).

PIA Primary Interop Assembly (Ensamblado Primario de Interoperabilidad).

PLN Procesamiento del Lenguaje Natural (del inglés, NLP Natural Language Processing).

PYMES Pequeñas Y Medianas Empresas.

SMP Symetric Multi-Processing (Multiproceso Simétrico).

SQL Structured Query Language (Lenguaje de Consulta Estructurado).

XML eXtensible Markup Language (Lenguage de Marcas Extensible).

REFERENCIAS

[1] *E.F. Codd. "A Relational Model for Large Shared Data Banks". Communications of the ACM, 1970.*

[2] *I.Androutsopoulos, G. Ritchie, and P. Thanisch, "Natural Language Interfaces to Databases An Introduction," Natural Language Engineering, Vol. 1, No. 1, 1995.*

[3] *E.F. Codd. "Seven Steps to RENDEZVOUZ with the casual user". NorthHolland Publishers, 1974.*
(http://www8.cs.umu.se/~mjm/pubs/vldb.pdf) Último acceso 05-05-2011.

[4] *J´a Te Digo — Uma interface em l´ıngua natural para uma base de dados de cinema.*
(http://www.inesc-id.pt/pt/indicadores/Ficheiros/3348.pdf). Último acceso 17-05-2011.

[5] *Dadiv H.D Warren and Fernando C.N Pereira. "An Efficient Easily Adaptable System for Interpreting Natural Language Queries".*
(http://acl.ldc.upenn.edu/J/J82/J82-3002.pdf). Último acceso 27-04-2011.

[6] *Gladys Rosalia Rocher Silva "Traducción de Queries en Prolog a SQL".*
(http://www.researchgate.net/publication/37613346_Traduccin_de_Queries_en_Prolog_a_SQL). Último acceso 01-06-2011.

[7] *Bruce W. Ballard. "User Specification of Syntactic Case Frames in TELI, A Transportable, User-Customized Natural Language Processor".). Published in "Proceeding COLING '86 Proceedings of the 11th conference on Computational linguistics."*
(http://delivery.acm.org/10.1145/1000000/991500/p454-ballard.pdf?ip=81.172.122.75&CFID=28616961&CFTOKEN=30276992&__acm__=1308043659_49337e40c4217775b054908df6319de8). Último acceso 16-05-2011.

[8] *Mrs. Neelu Nihalani, Dr. Sanjay Silakar, Dr. Mahesh Motwani. "Natural language Interface for Database: A Brief review". IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.*
(http://www.ijcsi.org/papers/IJCSI-8-2-600-608.pdf). Último acceso 11-05-2011.

[9] *I.Androusopoulos, G.D Ritchie, P. Thanisch. "Natural Language Interfaces to Databases – An Introduction".*
(http://arxiv.org/PS_cache/cmp-lg/pdf/9503/9503016v2.pdf). Último acceso 12-05-2011.

[10] *Evaristo Daniel Chay Coyoc, ITESM Campus Morelos.Mayo 1990. (http://w3.mor.itesm.mx/~jtorres/Datos.html). Último acceso 19-05-2011*

[11] *"Generator of Natural Language Databases Interfaces". Último acceso 22-05-2011. (http://www.vai.dia.fi.upm.es/inq/projects/paso.htm).*

[12] *Androusopoulos, G.D Ritchie, P. Thanisch. "Masque/SQL. An Efficient and Portable Natural Language Query Interface for Relational Databases".*

[13] *Nick Cercone, Paul McFetridge, Fred Popowich, Dan Fass, Chris Groeneboer, Gary Hall. "The SystemX Natural Language Interface:Design, Implementation and Evaluation". (http://www.google.es/url?sa=t&source=web&cd=3&ved=0CC0QFjAC&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.42.53%26rep%3Drep1%26type%3Dpdf&rct=j&q=The%20SystemX%2C%20Natural%20Language%20Interface%3A%20Design%2C%20%20Implementation%20and%20Evaluation&ei=YTD3TaDCDMqx8QOw6ZG8Cw&usg=AFQjCNF-3yNf2ujsBBKKcSCXWwy6H8m_mg&sig2=zXpk97XLzKHXpRjWJt5qcw). Último acceso 04-05-2011.*

[14] *Article "BIM: Testing the NLP waters." (http://www.lim.nl/monitor/bim.html). Último acceso 01-06-2011.*

[15] *INTELLECT. (http://domino.research.ibm.com/library/cyberdig.nsf/0/b4b11bbbffad08d3852571370058a6d3?OpenDocument).*

[16] *Juan José González Barbosa. "Traductor de Lenguaje Natural Español a SQL para un Sistema de Consultas a Bases de Datos". Último acceso 03-06-2011. (http://www.gelbukh.com/thesis/Juan%20Javier%20Gonzalez%20Barbosa%20-%20PhD.pdf).*

[17] *Q&A. (http://www.quickanswer.com/home.htm). Último acceso 03-06-2011.*

[18] *SQHAL. (http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura/). Último acceso 03-06-2011.*

[19] *EDITE. (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.890&rep=rep1&type=pdf). Último acceso 03-06-2011.*

[20] *Clara Bagnasco, Amedeo Cappelli, Bernardo Magnini. "TAMIC-P. A Dialogue Environment for Accessing Public Administration Data: the TAMI C-P System. (http://www.quinary.it/wp-content/uploads/2008/03/tamic99.pdf). Último acceso 03-06-2011.*

[21] *Ana M. Popescu, Oren Etzioni, Henry Kautz. "Towards a theory of natural language interfaces to databases," University of Washington, D.C. (http://www.cs.washington.edu/research/projects/webware1/www/precise/precise.html). Último acceso 22-05-2011.*

[22] *(http://www-nlpir.nist.gov/MINDS/FINAL/NLP.web.pdf). Último acceso 29-05-2011.*

[23] *(http://www.tamps.cinvestav.mx/sites/default/files/seminario.rodolfo.pazos_.pdf.zip) Último acceso 22-06-2011*

[24] *(http://www.marcocantu.com/edelphi/EssentialSQL_md6.pdf). Último acceso 24-06-2011.*

[25] *(http://zarza.usal.es/~fgarcia/docencia/poo/04-05/Trabajos/XAML.pdf). Último acceso 15-06-2011.*

[26] *Manual de Xaml . (http://www.megaupload.com/?c=premium&login=1). Último acceso 28-06-2011.*

[27] *VSTO. (http://invertedindex.wordpress.com/2009/06/10/vsto-how-to-create-a-toolbar-for-an-excel-add-in/). Último acceso 22-06-2011.*

[28] *Gestión de proyectos. (http://administracionelectronica.gob.es/recursos/pae_000001038.pdf). Último acceso 04-06-2011.*

[29] *Procesamiento del Lenguaje Natural. Escuela Superior de Ingeniería. Ingeniería. Ingeniería técnica en Informática de Gestión. Introducción a la Inteligencia Artificial. Depto. De Lenguajes y Sistemas Informáticos. Universidad de Cádiz. (http://webs.ono.com/usr008/iqpeblan/files/tema5a.pdf). Último acceso 22-05-2011.*

[30] *S. Whittaker y P. Stenton, "User Studies and the Design of Natural Language Systems", In Proceedings of the 4th Conference of the European Chapter of ACL, Manchester, England. Hewlett-Packard Laboratories, April 1989.*

[31] *"Spanish Natural Language Interface for a Relational Database Querying System". Lecture Notes in Artificial Intelligence (Text, Speech and Dialogue), Vol. 2448, ISSN 0302-9743, Springer-Verlag, Sep. 2002, pp. 123-130.*

[32] *"A Domain Independent Natural Language Interface to Databases Capable of Processing Complex Queries". Lecture Notes in Artificial Intelligence (MICAI 2005: Advances in Artificial Intelligence), ISSN 0302-9743, Springer-Verlag, 2005.*

APÉNDICES

APÉNDICE A

ESTUDIO DE VIABILIDAD DEL SISTEMA

RESUMEN

Este apéndice contiene el Estudio de Viabilidad del Sistema asociado a este proyecto.

El presente documento recoge el conjunto de requisitos identificados y establecidos para el desarrollo del traductor de sentencias NQPL. El propósito de este es la traducción de sentencias expresadas en lenguaje natural al lenguaje SQL, todo ello a través de un interfaz de interacción entre usuario y máquina.

HOJA DE ESTADO DEL DOCUMENTO

NQPL		
EVS. ESTADO DEL DOCUMENTO		
Versión	**Fecha**	**Motivación**
0.01	13-06-2011	Primera versión

EVS. Tabla 1. Formato hoja de estado del documento.

REGISTRO DE CAMBIOS

NQPL		
EVS. HISTORIAL DE CAMBIOS		
Versión	**Fecha**	**Motivación**
0.01	13-06-2011	Primera versión

EVS. Tabla 2. Formato historial de cambios.

NQPL			
EVS. VALIDACIÓN DEL DOCUMENTO			
Versión	**Fecha**	**Cargo**	**Nombre y Apellidos**
0.01	13-06-2011	Jefe de Proyecto	Carlos Cócera Pérez

EVS. Tabla 3. Formato validación del documento.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

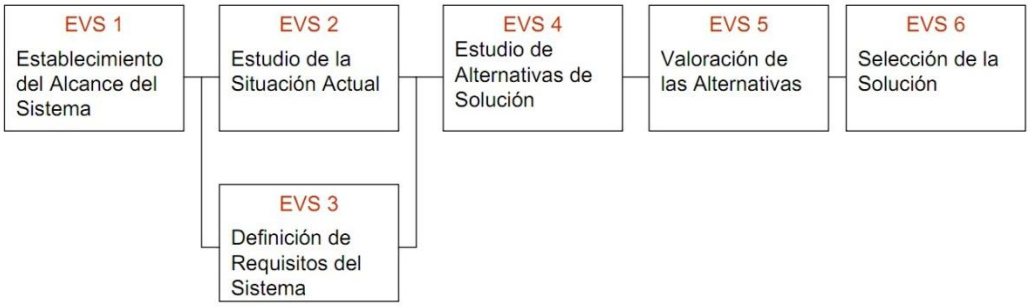
ÍNDICE DE TABLAS

1. DESCRIPCIÓN Y OBJETIVOS

¿Qué se entiende por viabilidad de un proyecto?

El grado de viabilidad de un proyecto es determinado por un conjunto de documentos que establecen si reúne las características, condiciones técnicas y operativas que aseguren el cumplimiento de las metas y objetivos definidos. Por tanto el Estudio de Viabilidad del Sistema consistirá en el aseguramiento de que un sistema cumpla con los objetivos preestablecidos durante su fase de desarrollo.

El Estudio de Viabilidad del Sistema está basado y es conforme con el estándar IEEE[18] 830-1998 y su objetivo consiste, por tanto, en el análisis de un conjunto concreto de necesidades de usuario para proponer una solución a corto plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas y resultará en un estudio que definirá al mismo.



EVS. Figura 1. Flujo de tareas del Estudio de Viabilidad del Sistema

Para ello, es necesario identificar los requisitos que se han de satisfacer y se estudia, si es preciso, la situación actual (*ver "3. ESTUDIO DE LA SITUACIÓN* ACTUAL" *y "4. DEFINICIÓN DE REQUISITOS DEL SISTEMA*").A partir del estado inicial, la situación actual y los requisitos recogidos, se han valorarán las alternativas de solución y optará por la más adecuada, definiendo y estableciendo a posteriori cuál será su planificación. A continuación se muestra una tabla que recoge las subtareas que se llevarán a cabo durante el proceso de definición del EVS:

[18] http://www.i333.org/.Último acceso 03-06-2011.

2. ESTABLECIMIENTO DEL ALCANCE DEL SISTEMA

Durante esta actividad se va a analizar el alcance del proyecto y a identificar las restricciones relativas a la sincronización con otros proyectos que puedan interferir en la planificación y futura puesta a punto del sistema objeto del estudio. Además, durante este proceso, se estudiará el plan de proyectos para determinar las posibles dependencias con otros proyectos. Una vez establecido el alcance, se identificarán las unidades organizativas afectadas por el sistema, así como su estructura y responsables de las mismas. Para determinar los responsables se tendrá en cuenta a quiénes afecta directamente y quiénes podrán influir en el éxito o fracaso del mismo.

2.1. ESTUDIO DE LA SOLICITUD

En esta tarea se van a describir las necesidades planteadas por el usuario de cara al sistema y qué tipo de restricciones de carácter económico, técnico, operativas y legales pueden afectar al mismo.

El presente proyecto está motivado por la necesidad de ofrecer a los usuarios de sistemas gestores de información una herramienta capaz de agilizar y facilitar la tarea de administrar información sin que el conocimiento técnico del lenguaje de manipulación de datos sea un requisito para dichos usuarios. El presente estudio busca determinar los diversos aspectos relacionados con la realización del proyecto que dé solución a la solicitud planteada. Es importante mencionar que no existen restricciones legales limitantes que pudieran afectar al desarrollo del proyecto.

2.2. IDENTIFICACIÓN DEL ALCANCE DEL SISTEMA

De acuerdo a la solicitud antes planteada y al contexto conceptual inicial establecido a lo largo del Capítulo I, se puede inferir que la herramienta será empleada principalmente en infraestructuras de empresa, no dejando a un lado un uso en sistemas más específicos. Dentro de la solicitud anteriormente planteada, se establece la siguiente lista de requerimientos relativos a las restricciones que intervienen en la delimitación del ámbito del sistema:

CONCEPTO	DESCRIPCIÓN
Geográfico	La aplicación podrá ser empleada en principio en organizaciones de habla hispana. En el futuro se integraran MUIs.
Tecnológico	La solución deberá adaptarse a una base de datos SQL y deberá funcionar al menos bajo Windows XP y Windows 7. En el futuro se planteará que el intérprete pueda traducir sentencias a otro tipo de lenguaje de manipulación de datos como Access. Para la integración como componente, será necesario disponer de la herramienta Microsoft Office (versiones 2007-2010).
Temporal	El proyecto deberá completarse a más tardar en Diciembre de 2011.
Operativo	El resultado del proyecto deberá aportar un grado de simplicidad mayor a las posibles herramientas existentes, tales como facilidad de uso de la interfaz, interpretación de los resultados obtenidos, etc.

EVS. Tabla 4. Alcance del Sistema.

Éste proyecto no forma parte de ningún plan de sistemas externo y no está ligado a las funciones de ninguna empresa. A demás de esto, no se contempla la integración del sistema final a ningún organismo ni departamento del cual pudiera llegar a depender el desarrollo del proyecto ni afectar al desarrollo de otros.

2.3. ESPECIFICACIÓN DEL ALCANCE DEL SISTEMA

De acuerdo a la información contenida en el Capítulo I y Capítulo II podría resultar innecesaria la evaluación de la situación actual. Sin embargo, es importante identificar las diferentes tecnologías existentes que resuelvan ya sea, parcial o totalmente, la solicitud antes mencionada y de ésta forma conocer el estado del arte actual para contrastar las posibles

soluciones, elegir aquella que cumpla con las restricciones y resolver la solicitud. Es también importante determinar la situación de los factores tecnológicos que están involucrados con los requisitos generales de modo que el equipo de análisis pueda hacer una consideración apropiada en el momento de la deliberación sobre qué solución se adapta más apropiadamente al problema planteado.

3. ESTUDIO DE LA SITUACIÓN ACTUAL

3.1. VALORACIÓN DEL ESTUDIO DE LA SITUACIÓN ACTUAL

En función de los objetivos establecidos para esta actividad y considerando la descripción general de la solución, se identifican las características tecnológicas del proyecto NQPL, que resulta necesario analizar con el fin de determinar el alcance del sistema actual. El estudio no requerirá un nivel considerable de profundidad, ya que solo se requiere una descripción acerca de los problemas de la tecnología actual y en algunos casos particulares que pudieran ser de interés para el análisis de la solución. Sin embargo, es necesario abordar de manera analítica la arquitectura de la tecnología y sistemas actuales.

3.2. IDENTIFICACIÓN DE LOS USUARIOS PARTICIPANTES EN EL ESTUDIO DE LA SITUACIÓN ACTUAL

Los usuarios o stakeholders participantes en el proyecto son todas aquellas personas a las que el proyecto involucra en cualquiera de las fases de su ciclo de vida.

Para la realización del estudio solo se requiere la participación de un miembro del equipo del presente proyecto; no es necesario involucrar personas externas ya que el proyecto no forma parte de ningún Plan de Sistemas de Información existente y tampoco de alguna empresa o departamento. Como resultado se puede declarar que todos los roles implicados directamente en el proceso del proyecto serán ejecutados por una única persona. A pesar de esto, se van a desglosar los roles participativos en dos grupos: implicados directos en el desarrollo del proyecto y los implicados indirectos.

PARTICIPANTES DIRECTOS	
PERFIL	**FUNCIONES**
Jefe de Proyecto	• Define el proyecto y evaluar sus necesidades. • Redacta las especificaciones del proyecto. • Calcula el costo del proyecto. • Contrata al equipo de producción. • Realiza un seguimiento e informes del progreso del proyecto, en términos de calidad, costo y plazos de entrega.
Analista	• Determina la mejor forma de resolver un problema de sistemas de información. • Reúne información y determina los requisitos y especificaciones del sistema. • Diseña el nuevo sistema.
Programador	• Traslada las especificaciones de la fase de diseño a código ejecutable.
Administrador de la Base de Datos	• Diseña y mantiene la Base de Datos

EVS. Tabla 5. Usuarios directos participantes en el proyecto.

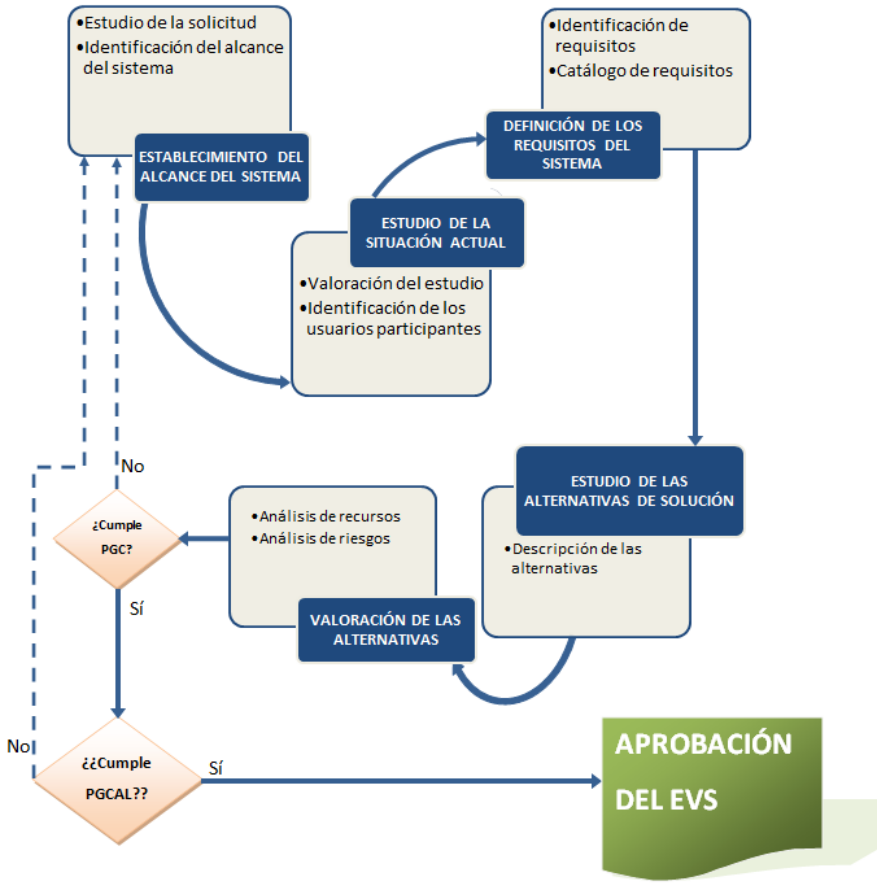
PARTICIPANTES INDIRECTOS	
PERFIL	**FUNCIONES**
Cliente	• Persona conocedora del mundo del desarrollo de software. Es el encargado de que el proyecto satisfaga los requisitos que desea para el producto final.
Usuarios finales	• Son aquellos que harán uso del producto final.

EVS. Tabla 6. Usuarios indirectos participantes en el proyecto.

3.3. PLAN DE TRABAJO

El plan de trabajo es el documento de planificación y gestión que determina y proporciona las condiciones para planificar el trabajo, en este caso, del Estudio de Viabilidad del Sistema. El objetivo consistirá en identificar los problemas a solucionar, convertirlos en objetivos precisos y verificables, indicando los recursos necesarios y los obstáculos a neutralizar, adoptando una estrategia y ejecutando las acciones necesarias para conseguir dichos objetivos y obtener resultados que satisfagan al cliente.

Como se ha mencionado anteriormente, será una única persona quien desempeñe la función de los roles implicados directamente con el proyecto, en el EVS en este caso en concreto. A continuación se muestra un diagrama de flujo de etapas del EVS.



EVS. Figura 2. Flujo actividades del proceso del EVS.

3.4. DESCRIPCIÓN DE LOS SISTEMAS DE INFORMACIÓN EXISTENTES

INTRODUCCIÓN

En esta tarea se lleva a cabo el estudio de los sistemas de información actuales afectados por el PSI. Para cada sistema de información se recogen, al menos, las características básicas relativas a datos, software de aplicación, procesos de la organización a los que da soporte y de qué forma lo hace, flexibilidad, carencias, riesgos y posibles amenazas. En función del tipo de sistema de información y de los objetivos de su estudio se recopila además, para cada uno de ellos, información procedente de diversos puntos de vista (la opinión de usuarios de los sistemas de información, de analistas de desarrollo, de personal de operación, etc.).

SISTEMAS ILNBD

La Generación del Lenguaje Natural (GLN) es una subárea de la inteligencia artificial y de la lingüística computacional que concierne con la construcción de sistemas de la computadora que puede producir textos entendibles en español u otros idiomas humanos o tal vez una representación subyacente no-lingüística de información. Los sistemas de generación del lenguaje natural combinan conocimiento sobre el lenguaje y la aplicación. Podemos agrupar los sistemas existentes en relación a la técnica que emplean para traducir el lenguaje natural a un lenguaje artificial:

- **Emparejamiento de patrones**: Algunas de las primeras ILNBD trataban con técnicas de emparejamiento de patrones para resolver las consultas de los usuarios. Otro punto es que lo sistemas de emparejamiento de patrones a menudo devuelven un tipo de respuesta aun cuando la entrada esta fuera del rango de las frases que fueron pensadas para este tipo de sistemas.

  El análisis elaborado y los módulos de interpretación no son necesarios y estos sistemas son fáciles de implementar.

- **Sistemas basados en sintaxis**: En los sistemas basados en sintaxis la consulta del usuario es analizada (análisis sintáctico), y el árbol de análisis resultado de este proceso es directamente transformado a una expresión de consulta en algún lenguaje formal. El problema es que estos árboles no son válidos para una plataforma multilenguaje ya que, sintácticamente, las expresiones en distintos idiomas pueden ser distintas. Este es el método más aceptable, ya que, independientemente del lenguaje, se podrían generar módulos de traducción dependientes de la *CultureInfo.CurrentCulture* de la aplicación.
- **Sistemas basados en gramáticas semánticas**: En los sistemas de gramáticas semánticas, la función de pregunta-respuesta es hecha por el análisis de la entrada y el mapeo del árbol de análisis es transformado a una expresión de consulta en algún lenguaje formal. La diferencia con el anterior son las categorías de la gramática que no necesariamente corresponden a conceptos de sintaxis. Construir este tipo de sistemas supone un gran esfuerzo debido a la complejidad del lenguaje a traducir.
- **Lenguajes intermedios de representación**: La mayoría de las ILNBD actuales primero transforman la consulta en lenguaje natural en una consulta lógica intermedia, expresada en algún lenguaje. La consulta lógica intermedia expresa el significado de la consulta del usuario en términos lógicos de alto nivel, que son independientes de la estructura de la base de datos y del lenguaje mismo. La consulta lógica es entonces traducida a una expresión de consulta en algún lenguaje formal de consulta de base de datos evaluado. Actualmente los sistemas modernos de lenguaje natural usan varias representaciones intermedias y no se limitan a una sola.

La característica principal de los sistemas de comprensión del lenguaje natural es que ellos procesan representaciones del significado de las oraciones y usan estas representaciones en procesos de razonamiento. Se mencionaron tres niveles de representación, procesamiento sintáctico que trata con las propiedades estructurales de las oraciones, procesamiento semántico se encarga de la forma lógica que representa el significado libre de contexto de la oración y procesamiento contextual que conecta el lenguaje con el dominio de la aplicación.

En general, el panorama del procesamiento del lenguaje natural es una tarea compleja que requiere el desarrollo de un buen número de recursos. Destacamos los siguientes:

- Un depósito de información lingüística con un rico y bien estructurado componente léxico.
- Sistemas de representación apropiados.
- Gramáticas versátiles.

Los problemas de implantación y la demanda extraordinaria en recursos de cómputo han limitado el desarrollo de las interfaces en lenguaje natural. Sin embargo, numerosos programadores e investigadores están trabajando de manera activa en el desarrollo de interfaces en lenguaje natural, de tal forma que es un área que definitivamente continuará creciendo y por ello requerirá de un estrecho seguimiento.

Sin embargo, los problemas más importantes que este enfoque presenta es la elección del vocabulario de la interlingua (Arnold et al. 1994), es decir, los conceptos primitivos de la representación del significado. Por ejemplo, se puede elegir un concepto corner para hacer referencia a la palabra inglesa corner; esto puede parecer natural a un hablante de la lengua inglesa, pero no a un hablante de español, que contempla dos vocablos distintos según la orientación: esquina y rincón. Entonces deberíamos contemplar dos conceptos primitivos distintos: inside-corner y outside-corner. En ruso no existe ninguna traducción para azul, sino que existe goluboi (azul claro) y sinii (azul oscuro), ¿cuál sería la representación de interlingua de azul?

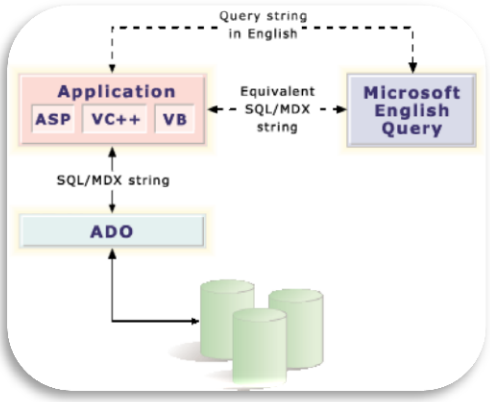
Las arquitecturas descritas anteriormente, permiten a los usuarios interactuar solamente con una base de datos. ¡En algunos casos es necesario el acceso a múltiples aplicaciones! Tal vez la información necesaria para contestar la consulta del usuario se encuentre distribuida en múltiples bases de datos heterogéneas. Algunas ILNBD permiten al usuario acceder de manera transparente múltiples bases de datos a través de una simple consulta en algún lenguaje. La ILNBD traduce la consulta lógica en un conjunto de consultas de bases de datos, cada una expresada en el lenguaje de consulta

correspondiente al SGBD usado. La ILNBD recoge y une los resultados parciales regresados por cada SGBD, y reporta la respuesta al usuario.

Es preciso realizar un estudio de los sistemas de información actuales que puedan ser potencialmente competentes frente al sistema que se está desarrollando y recoger, al menos, las características básicas relativas a datos, software de aplicación, procesos de la organización a los que da soporte y de qué forma lo hace, flexibilidad, carencias, riesgos y posibles amenazas.

MICROSOFT ENGLISH QUERY (MSEQ)

Microsoft English Query es un componente de SQL Server 7.0 que provee a los usuarios para realizar consultas a la base de datos usando inglés a un nivel más alto que el de la programación común. Para implementar la búsqueda en lenguaje natural, primero se utiliza una herramienta para relacionar las entidades de la base de datos con los objetos del dominio. Estas relaciones se usan para desarrollar el análisis de las consultas de los usuarios, el cual produce mejores resultados de los que se obtendrían usando solo tecnología basada en palabras claves. El sistema de ejecución puede ser incluido con COM soportando ambientes como Visual C++, Visual Basic y ASP.



EVS. Figura 3. Arquitectura MSEQ.

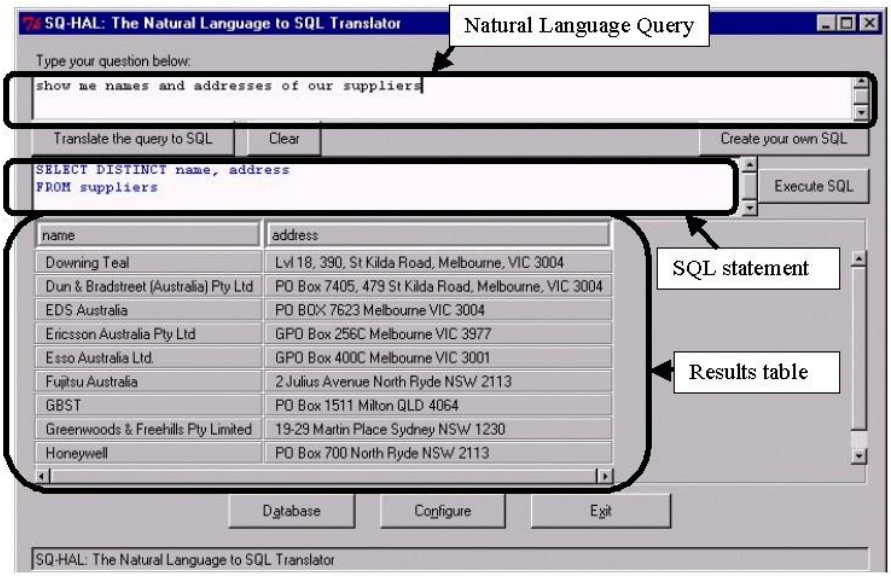
Esto habilitaría que las consultas en inglés puedan ser incrustadas en software creado por los clientes así como para sitios que soporten ASP.

El inconveniente de este sistema es que English Query solo está disponible para plataformas Win32 y soporta solo fuentes de datos que tengan OLEDB tales como Oracle y Microsoft Access, lo cual impide satisfacer la multiplataforma. Además, es un sistema de traducción basado en gramáticas semánticas que es bastante complejo.

SQ-HAL

SQ-HAL está dirigido a la reducción de la complejidad en las consultas de base de datos. Primero es necesario usar un lenguaje que sea entendido por cualquiera, ya sea un experto programador de bases de datos o una persona sin conocimiento alguno sobre computadoras. Para realizar la traducción del lenguaje natural, SQ-HAL necesita conocer la arquitectura de la base de datos. Este proceso es automatizado tanto como ha sido posible para minimizar la complejidad para el usuario.

Los privilegios de acceso a la base de datos son diferentes para cada usuario. Por tanto el acceso multiusuario es soportado por SQ-HAL. Solo las consultas son implementadas en SQ-HAL y características de bases de datos tales como agregar, borrar o actualizar los datos situados en la base de datos no son considerados.

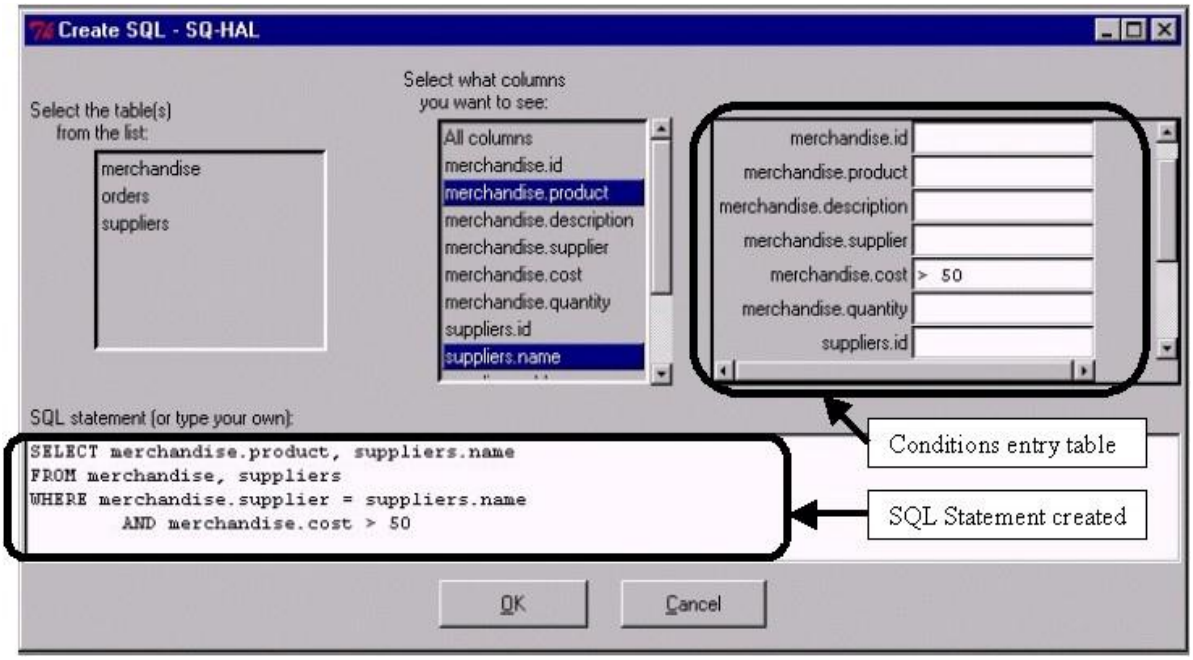


EVS. Figura 4. Interfaz principal de SQ-HAL.

SQ-HAL es implementado como software de computadora. Este no sólo traduce la consulta en LN a expresiones de SQL, sino que además las ejecuta y recupera la

información de la base de datos y despliega ésta al usuario. La interfaz de usuarios también es implementada haciendo más fácil para los usuarios el uso de este programa. SQ-HAL es implementado como una base de datos y plataforma independiente ajustándose a las diferencias entre varios sistemas de base de datos.

Además, permite realizar las consultas SQL sin usar el lenguaje natural, para aquellos usuarios avanzados que no requieran de esa funcionalidad.



EVS. Figura 5. Interfaz creación de sentencias con SQ-HAL.

SQ-HAL traduce oraciones en inglés, frases y palabras clave expresiones en SQL. También está equipado con capacidades de aprendizaje para las gramáticas que no sea capaz de entender. El problema que se detecta en este sistema es que trabaja en inglés y, de nuevo, funciona únicamente en plataformas Win32 y que hay que conocer previamente la arquitectura de datos sobre la cual se va a trabajar. Además, traduce mediante análisis de gramáticas semánticas, resultando un sistema complejo de codificar.

En cuanto a la interfaz, es muy sencilla pero podría mejorarse visualmente mediante XAML y WPF.

En cuanto a los dos sistemas, no están integrados como componentes en ninguna suite, son aplicaciones independientes lo cual va a requerir que el usuario no pueda ejecutarlas, por ejemplo, desde Excel y tenga que tener instalaciones independientes.

3.5. RESTRICCIONES DEL PROYECTO

En este apartado se pretende definir qué restricciones afectan al desarrollo de este proyecto, de qué limitaciones consta teniendo en cuenta los recursos de que se dispone para la consecución del mismo. Se pueden distinguir 6 categorías restrictivas:

Metodologías de Desarrollo: Se ha optado por Métrica 3 ya que el encargado de aplicar y definir las tareas que se van a llevar a cabo durante la fase de planificación, que es competencia del Jefe de Proyecto, únicamente está familiarizado con esta metodología, desconociendo otras metodologías como Rational Unified Process[19] (RUP) o Booch. Tiene conocimientos de "SCRUM y Metodologías Ágiles" pero no los suficientes como para aplicarla.

Lenguajes de Programación: Dado que el programador posee más experiencia en entornos de desarrollo .NET con C#, se ha optado por no emplear otros lenguajes como J# o C++ ya que exigiría un periodo de formación y retrasaría la planificación.

Requisitos de Software Desarrollo: En la empresa, se han instalado equipos que cumplen con los requisitos de hardware recomendados.

- Visual Studio es una herramienta desarrollada por Microsoft y como tal, únicamente está preparada para ejecutarse en los sistemas operativos Windows XP, Windows Vista y Windows 7. Esto supone una restricción multiplataforma.
- Para el desarrollo del proyecto se han utilizado todos los recursos económicos en adquirir un equipo que cumpla con los requisitos de hardware mínimos y para comprar originales de Visual Studio 2010 Ultimate Edition y Windows 7.

[19] http://www.ibm.com/software/awdtools/rup/.Último acceso 01-06-2011.

El resto del software es libre como Firebird, ya que no se dispone suficientes fondos para adquirir SQL Server 2008 u otro software de pago.

Restricciones de Hardware: Para poder ejecutar Visual Studio 2010 serían:

REQUISITOS DE HARDWARE MÍNIMOS	REQUISITOS DE HARDWARE RECOMENDADOS
Equipo con un procesador de 1,0 GHz	Equipo con un procesador de al menos 1,6 ghz
512 MB de RAM	1024 MB de RAM
Espacio disponible en disco duro de 3 GB	Espacio disponible en disco duro de 5,1 GB
Unidad de disco duro de 5.400 rpm	
.NET Framework 3.5 o superior	.NET Framework 4.0
Tarjeta de vídeo compatible con DirectX	

EVS. Tabla 7. Requisitos de Software.

Requisitos de Software Usuario: El cliente ha establecido que la plataforma de desarrollo, inicialmente y previo a posibles procesos de portabilidad, sea cualquier sistema Microsoft Windows en versiones superiores al XP (a excepción de Windows ME y Windows 2000).

- Es necesario tener instalados Windows XP, Server 2003, Server 2008, Vista o Windows 7 para poder ejecutar la aplicación y tener instalado .NET 3.5, ya que versiones anteriores no admiten características de la aplicación, como el lanzamiento a través del Ribbon de Office 2007 o 2010.

Restricciones de implementación: Debido a la escasez de recursos humanos y de tiempo para el desarrollo del proyecto se ha detectado que el inconveniente de implementar un módulo de traducción compleja ya que podría exceder la planificación de tiempo y por tanto de inversión económica. Por ello se ha optado por la opción menos complicada dentro de los sistemas de traducción y análisis descritos en capítulos anteriores.

Restricciones de codificación: El código XAML quitará tanta codificación como sea posible para quitarle funcionalidad a C#.

Se tratará de emplear las herramientas de Microsoft más estandarizadas para conseguir que NQPL sea lo más estandarizado posible (Visual Studio, .NET, Windows 7).

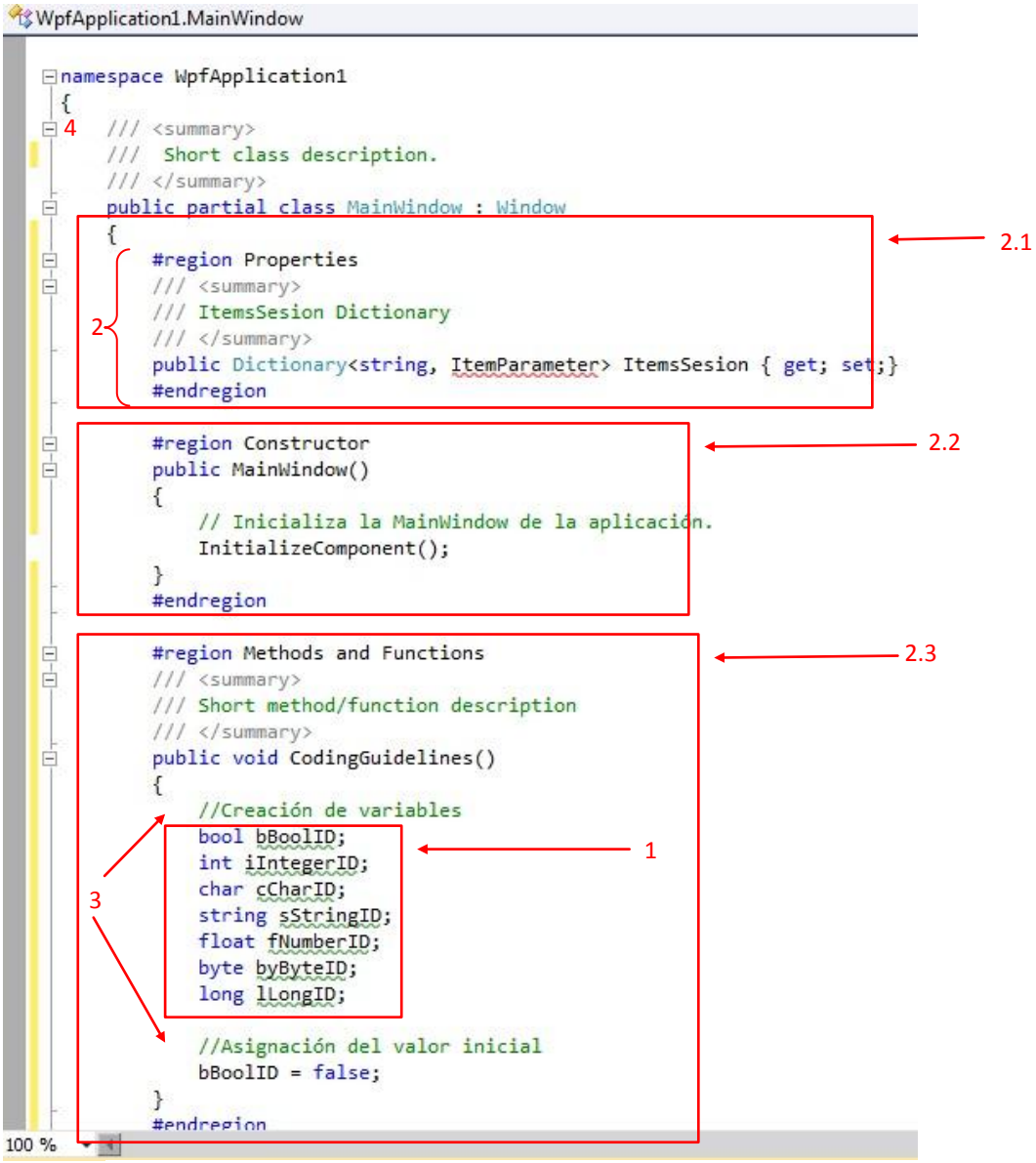
A la hora de implementar código, se seguirán unos Coding Guidelines. Esto implica determinar un convenio a la hora de nombrar elementos del código. Por ejemplo, las variables deberán nombrarse con un identificador que comenzará por letra capital y estará precedido por un prefijo que indicará el tiempo de variable de qué se trata. EL formato se corresponderá al siguiente:

<identificadorTipoDato><NombreVariable>[1]

Por ejemplo, una variable de tipo booleano podría ser *bIsFalse*, una de tipo entero sería *iHeight*, etc. Los tipos de variables que se regirán por esta nomenglatura se pueden ver en "*EVS. Figura 6*". Además, toda la aplicación estará íntegramente en inglés, tanto nombres de variables, de clases, assemblies, etc., como *<remarks>* y *<summary>* y comentarios.

Creación de regiones de código: La estructura de cada clase deberá tener claramente diferenciados bloques de código mediante la marca *#region*[2] (*ver "EVS. Figura 6"*).Se empleará mínimo para marcar los siguientes bloques:

- Bloque de declaraciones[2.1].
- Constructor[2.2].
- Bloque de métodos y funciones[2.3].



EVS. Figura 6. Coding Guidelines en Visual Studio 2010.

A criterio del programador, se podrán añadir otras marcas de bloque. El resto de líneas de comentarios[3] que se hagan acerca del código se harán en castellano. Como última regla de codificación, en el bloque de declaración de propiedades de la clase, habrá que describir con la marca *<summary>*[4] cada una de ellas.

El propósito de todo esto es hacer que el código sea lo más inteligible posible para nuevas incorporaciones al equipo de desarrollo y agilizar su familiarización con le.

4. DEFINICIÓN DE REQUISITOS DEL SISTEMA

Esta actividad incluye la determinación de los requisitos generales, mediante una serie de sesiones de trabajo con los usuarios participantes, que hay que planificar y realizar. Una vez finalizadas, se analiza la información obtenida definiendo los requisitos y sus prioridades, que se añaden al catálogo de requisitos que servirá para el estudio y valoración de las distintas alternativas de solución que se propongan. El cumplimiento de requisitos determinará el grado de satisfacción del cliente con el sistema.

Los requisitos tendrán una plantilla de presentación, a modo de tabla, que estará estandarizada y que recogerá los diferentes atributos de cada uno de ellos. Los distintos atributos, su significado y los valores que podrán admitir se regirán por las siguientes reglas:

Identificador: Sirve para referenciar a cada requisito unívocamente. Obedecen a la siguiente regla de codificación, donde RU indica que se trata de un requisito de usuario:

RU<tipo>-<Numeración>

Donde el tipo podrá ser C o R, cuando se trate de un requisito de capacidad o de restricción respectivamente.

Descripción: Se deberá dar una breve descripción del requisito.

Prioridad: Grado de importancia y urgencia en la implementación del requisito. Los posibles valores que puede tomar son: *Por defecto, Alta, Media o Baja.*

Estabilidad: Mide el grado en el que el requisito es susceptible a cambiar, o no, a lo largo de la vida del proyecto. Puede tomar los valores *Alta, Media o Baja.*

Fuente: Indica la procedencia del requisito y tiene como objetivo dar información de cuál es la fuente para poder acceder a una información acerca del mismo más amplia. Los valores que puede tomar son:

- Desarrollo*:* El requisito ha sido puesto por el equipo de desarrollo, más en concreto por el Analista Funcional.
- Cliente*:* El requisito ha sido puesto por el equipo de UC3M[20].

Claridad: Determina la falta de ambigüedad del requisito. Puede tomar los valores *Alta, Media o Baja.*

Verificabilidad: Mide cuán fácil se puede comprobar la incorporación del requisito en el sistema. Puede tomar los valores *Alta, Media o Baja.*

4.1. REQUISITOS DE CAPACIDAD

RUC-001

Descripción: El sistema deberá permitir realizar consultas simples en lenguaje natural y traducirlas en sentencias de consulta en lenguaje SQL.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

[20] http://www.uc3m.es/. Último acceso 01-07-2011

RUC-002

Descripción: El sistema deberá permitir realizar operaciones simples en lenguaje SQL.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-003

Descripción: La aplicación deberá ser capaz de realizar consultas a la BD.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-004

Descripción: La aplicación deberá ser capaz de insertar campos en las tablas.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-005

Descripción: El sistema permitirá cambiar el idioma de inglés a español.

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-006

Descripción: La aplicación deberá mostrar el resultado de las operaciones de consulta en la ventana de aplicación.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-007

Descripción: La aplicación deberá ser capaz de borrar.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-008

Descripción: El sistema permitirá exportar el resultado a un documento XML.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-009

Descripción: El sistema permitirá cambiar el idioma de español a inglés.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-011

Descripción: El sistema permitirá borrar tablas en la base de datos (Drop).

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-012

Descripción: El sistema permitirá exportar el resultado a un archivo Excel 2010.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-013

Descripción: El sistema permitirá exportar el resultado a un archivo Excel 2003.

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-014

Descripción: El sistema permitirá borrar tablas (Drop).

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-015

Descripción: El sistema permitirá limpiar tablas (Truncate).

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-016

Descripción: El sistema permitirá exportar el resultado a una hoja de Excel.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-017

Descripción: La aplicación deberá poder ejecutarse desde un componente Office, a través del add-in registrado.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-018

Descripción: La exportación de datos deberá estar debidamente formateada.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUC-019

Descripción: Se podrá añadir valores de campos o columnas del visor de tablas a los editores de sentencias mediante un clic derecho de ratón.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

4.2. REQUISITOS DE RESTRICCIÓN

A continuación se expondrán los requisitos de restricción obtenidos del apartado 3.5:

RUR-001

Descripción: La metodología aplicada será Métrica versión 3.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-002

Descripción: El entorno de desarrollo será NET 3.5 o superior.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-003

Descripción: El lenguaje de programación empleado será C#.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-004

Descripción: El entorno visual se implementará mediante XAML y WPF.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-005

Descripción: Se cargará tanta funcionalidad de C# como sea posible en XAML.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-006

Descripción: El usuario de la aplicación deberá disponer de la computadora y el hardware mínimos para poder utilizarla.

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUR-007

Descripción: El usuario de la aplicación deberá disponer del sistema operativo Windows XP o Server 2003 y Microsoft office 2007 como mínimo para ejecutar la aplicación.

Prioridad: Alta.

Estabilidad: Baja.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUR-008

Descripción: La aplicación tendrá un módulo de traducción no complejo.

Prioridad: Alta.

Estabilidad: Baja.

Fuente: Desarrollo.

Claridad: Baja.

Verificabilidad: Media.

RUR-009

Descripción: Todo el proceso de codificación seguirá el estándar definido en los Coding Guidelines.

Prioridad: Media.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Media.

RUR-010

Descripción: Las representaciones en XAML seguirán recomendaciones definidas por la W3C[21] y se ceñirá lo máximo posible a la GUI publicada por Microsoft en MSDN acerca de estándares de diseño de GUIs.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

[21] http:\\www.w3c.org/. Último acceso 04-06-2010.

Verificabilidad: Media.

RUR-011

Descripción: La aplicación podrá ser utilizada por cualquier usuario no experto.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUR-012

Descripción: Los usuarios de la aplicación no necesitarán de un periodo de formación para utilizarla.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUR-013

Descripción: El sistema de ficheros deberá estar controlado por Sourcesafe.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-014

Descripción: El entorno de ejecución deberá cumplir con los requisitos de hardware mínimos para ejecutar la aplicación.

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RUR-015

Descripción: Las sentencias en lenguaje natural deberán finalizar con ".".

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-016

Descripción: Las sentencias con condiciones múltiples deberán separarse con la palabra "además". Esto se debe a que separarlas con "y" únicamente no es factible para el procesado.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-017

Descripción: Los operadores “>==”, “<==” y “!>” deberán expresarse con dichos operadores, no podrán expresarse con palabras como “mayor que”, “menor que”…

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-018

Descripción: No se podrán usar UNION, HAVING COUNT, GROUP BY.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RUR-019

Descripción: No se podrán hacer selects anidados.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

5. ESTUDIO DE LAS ALTERNATIVAS DE SOLUCIÓN

Este apartado se centra en la propuesta de otras alternativas que respondan positivamente a los requisitos planteados, considerando también los resultados obtenidos en el apartado “

3. ESTUDIO DE LA SITUACIÓN ACTUAL" del presente capítulo.

En la descripción de las distintas alternativas de solución propuestas, se debe especificar si alguna de ellas está basada, total o parcialmente, en un producto existente en el mercado.

5.1. DESCRIPCIÓN DE LAS ALTERNATIVAS DE SOLUCIÓN

Se procede a continuación a hacer un resumen de las alternativas seleccionadas para el estudio y de sus principales características.

COBERTURA	SQ-HAL	MICROSOFT ENGLISH QUERY
Plataforma	Win32, Linux.	Win32.
Lenguaje	Perl.	Visual Basic, C++.
Data Source	DBI.	OLEDB provider.
SGBD	Multiplex, Proxy, MySQL.	Sql Server.
Interfaz	Tk-Module.	Win32 GUI.
Idioma	Inglés.	Inglés.
Ventajas/Desventajas	• Parsing lento en Win32. • No soporta ADO. • Idioma.	• Solo Win32. • Calidad-precio. • Consumo de ram. • Idioma.

EVS. Tabla 8. Alternativas a NQPL.

Es indiscutible que Microsoft Windows es el sistema operativo más popular y con mayor número de usuarios del planeta. Aunque no es aceptado por todos, Win32 abarca el mayor número de proyectos de desarrollo de software y volumen de ventas. Pero hay que tener en cuenta Linux, sistema operativo que empieza a hacerse famoso entre los usuarios por su modularidad y capacidad de adaptación; más aún ahora que se han implementado distros con un shell que facilita a los usuarios de Windows la migración a estos sistemas basados en GNU/Linux. Sin embargo, diseñar aplicaciones hoy en día basadas en Perl, sin soporte Web y

con una interfaz de diseño obsoleto, inclina la balanza claramente hacia MSEQ. Hay que hacer hincapié en que, a ambos sistemas, les beneficiaría liberar el código. Esto supondría que numerosos desarrolladores podrían unir esfuerzos en corregir fallos, mejorar funcionalidad y acercar el producto a un mayor número de clientes, ¡y todo de manera gratuita!

La alternativa anteriormente planteada, dado que son sistemas bastante antiguos en relación a las herramientas de que disponemos en la actualidad, se podría "actualizar" del siguiente modo:

ALTERNATIVA 1

COBERTURA	NQPL
Plataforma	Win32 (Linux, Mac, etc con Mono).
Lenguaje	C#.
Data Source	ADO.NET.
SGBD	Firebird.
Interfaz	WPF y XAML.
Idioma	Multilenguaje.
Ventajas/Desventajas	Las descritas en el Capítulo II.

EVS. Tabla 9. Alternativa NQPL.

Otra alternativa que no está reflejada en este documento, pues no se ha encontrado ninguna aplicación similar, sería la siguiente:

ALTERNATIVA 2

COBERTURA	NJQPL
Plataforma	Win32, Linux, Mac OS, etc.

Lenguaje	Java.
Data Source	JDBC, JODB, etc.
SGBD	SQLYog, MySql, Ozone, Apache Derby, etc.
Interfaz	AWT, SWING.
Idioma	Multilenguaje.
Ventajas/Desventajas	Las comentadas en apartados anteriores

EVS. Tabla 10. Alternativa Opensource.

Esta sería una de las soluciones óptimas, un sistema multiplataforma, en lenguaje Java y con todas las herramientas de desarrollo y tecnologías libres. Pero debido a la aparición de Mono, sistema que permite portar aplicaciones de entorno .NET a otros sistemas operativos (de modo parecido a como lo hace Wine en Linux con aplicaciones Win32), podemos pensar en .NET como una solución óptima, y liberar código.

6. VALORACIÓN DE ALTERNATIVAS

Una vez definidas las alternativas, se va a proceder a realizar una valoración de las mismas, considerando el impacto en la organización, tanto tecnológico y operacional como organizativo. Además, se expondrán las expectativas en cuanto a los beneficios que se esperan contrastados con los costes asociados al desarrollo de las alternativas. Se finalizará analizando cuantitativamente los recursos y plazos exactos para planificar la alternativa.

6.1. IMPACTO TECNOLÓGICO, OPERACIONAL Y ORGANIZATIVO

De cara a afrontar cualquier cambio asociado a las posibles nuevas propuestas de mejora y optimización de los sistemas y tecnologías de información, es necesario realizar un estudio que tenga en cuenta las consecuencias que dichos cambios pueden acarrarle a la organización.

Entre los parámetros más destacados están: complejidad, rechazo cultural, coste, periodo de adaptación y miedo al cambio.

- **Complejidad:** Hace referencia a la dificultad que pueda tener la nueva tecnología en instaurarse en la compañía, lo cual es más que probable que exija inversiones en formación adaptativa.

  Si se observan las dos alternativas, la segunda supondría una inversión en formación, ya que el personal no está acostumbrado a trabajar con dichas tecnologías y habría que dar cursos de Java, Swing, probablemente Hibernate y JQuery.

- **Rechazo cultural:** No se refleja este factor en ninguna de las alternativas.
- **Coste**: En cuanto al coste, hay que reconocer que la segunda alternativa supondría un gran ahorro de inversión, ya que todas las herramientas necesarias son gratuitas, pero no olvidemos la inversión de cursos de formación que implica.

- **Periodo de adaptación**: Puede que el tiempo que tarde la compañía en adaptarse a la nueva tecnología.
- **Miedo al cambio**: En ocasiones las empresas tienen dudas acerca de la fiabilidad de las nuevas tecnologías, sobre todo cuando acaban de saltar al mercado. Por ello mantienen siempre un margen temporal, un umbral de cambio, que varía en función de la necesidad que tengan de adoptar la recién nacida tecnología y del beneficio-riesgo que suponga adoptarla.

6.1. ANÁLISIS DE LA INVERSIÓN DE RECURSOS

En este apartado se van a considerar las diferencias en términos de coste de recursos entre las alternativas propuestas. Se analizará exhaustivamente cada una de ellas, exponiendo las ventajas y desventajas principales que presenta cada una de ellas.

Para conseguir un estudio más concreto acerca de las aportaciones de cada una de las alternativas, se va a realizar un análisis ponderado de ciertos factores considerados fundamentales en el desarrollo de un producto de software. Los factores se ponderarán con valores de 0 a 10 y son los siguientes:

- **Coste del desarrollo:** Suma del total de gastos que ha supuesto que el desarrollo de la aplicación. Por encima del 5 pondera negativamente.
- **Versatilidad del producto:** Capacidad del producto de adaptarse a los cambios.
- **Portabilidad:** Capacidad del producto para funcionar correctamente en otras plataformas.
- **Tiempo de desarrollo:** Tiempo invertido en el desarrollo del software. Por encima del 5 pondera negativamente.
- **Factor de estandarización:** Grado de fidelidad a los estándares actuales.

Los valores que estén ponderados en negativo restarán puntos a la valoración global. A continuación se muestra la tabla de valoración de los factores para cada una de las alternativas propuestas:

FACTOR	ALT 1	ALT2	DESCRIPCIÓN
Coste	7	6	La alternativa 1 requiere de licencias y equipos algo más potentes que la alternativa 2 que emplea software gratuito. Sin embargo, la alternativa 2 requiere de cursos de formación que incrementan los gastos.
Versatilidad	7	7	Ambas alternativas emplean tecnologías que son capaces de adaptarse con rapidez a demandas de mercado.
Portabilidad	7	7	La alternativa 1, hasta la aparición de Mono y Wine presentaba rigidez a la hora de dotar a sus productos de la capacidad de ser portable. Ahora ambas tienen la misma capacidad.
Tiempo	6	9	La alternativa 2 implica una mayor inversión de tiempo debido a la necesidad de formar al equipo de desarrollo en el empleo de las nuevas herramientas (Java, Swing, Hibernate etc.) y del periodo de adaptación a las mismas.
Estandarización	9	9	Ambas se han ajustado lo máximo posible a los estándares, tanto en diseño, como en implementación.
Fiabilidad	9	6	Las herramientas de pago dan más seguridad a los usuarios que el entorno gratuito.

EVS. Tabla 11. Comparativa factores alternativas solución.

Como resultado la alternativa 1 ha obtenido un valor de 19 mientras que la alternativa 2 ha obtenido un 14. A partir de aquí se nos plantea la siguiente pregunta:

¿Por qué no utilizar lo mejor de ambas?

6.3. PLANIFICACIÓN DE ALTERNATIVAS

A partir de este análisis, se llega a la conclusión de que la primera alternativa es más fiable, da más sensación de seguridad y estabilidad y el tiempo de desarrollo es bastante menor. Pero los costes son mucho más altos. Como respuesta a la pregunta formulada a raíz de este análisis, podemos tomar la alternativa 1 pero intentando utilizar el mayor software libre para el desarrollo, siempre y cuando no se pierda fiabilidad.

La solución consistiría en conservar el entorno de desarrollo .NET 3.x, el Sistema operativo y las herramientas Visual Studio 2010 adquiridas previamente, C#, WPF y XAML, pero cambiar un SGBD comercial, como Oracle, por uno gratuito como Firebird o MySQL.

7. SELECCIÓN DE LA SOLUCIÓN

Se ha optado por la alternativa 1, con una leve modificación justificada, como opción óptima de acuerdo a los factores analizados y que influyen directamente en el desarrollo del mismo. En los siguientes apartados, se profundizará en la alternativa seleccionada con el fin de detallar aspectos técnicos más específicos que y que influirán en el desarrollo de la misma.

7.1. DESCRIPCIÓN DE LA SOLUCIÓN

Para el proyecto NQPL se ha hecho una fuerte apuesta por la popularidad del producto y gestión de tiempo, lo que va a permitir que el desarrollo de la solución entre en la planificación de tiempo y que el producto final tenga una sólida fiabilidad y confiabilidad de cara al usuario.

- **Microsoft Windows:** El corazón de cualquier máquina es el Sistema Operativo. El sistema menos estable en el mundo, en peligro constante de nuevas amenazas, vulnerable a casi cualquier ataque, se traba o cuelga constantemente, en cierta forma trata al usuario como un tonto, eso me hace recordad una frase que dice:

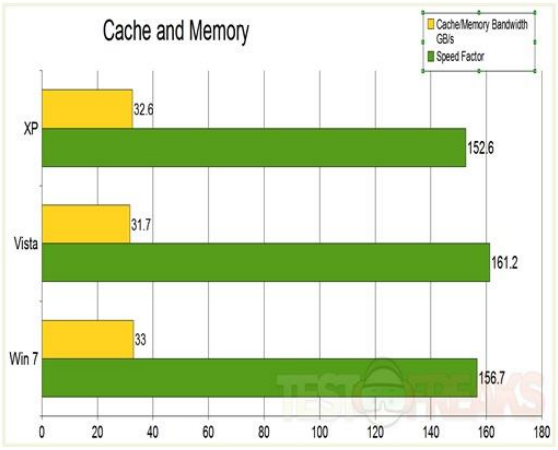
  "Si Microsoft fabricara coches, sustituiría la alerta de aceite, combustible, agua, etc. por un simple, error grave del sistema, hay que reiniciar"

  Sin embargo, la popularidad de Windows hace que la mayoría de las aplicaciones y juegos estén diseñados exclusivamente para este sistema. Ahora queda decidir, ¿Windows XP, Vista o 7?

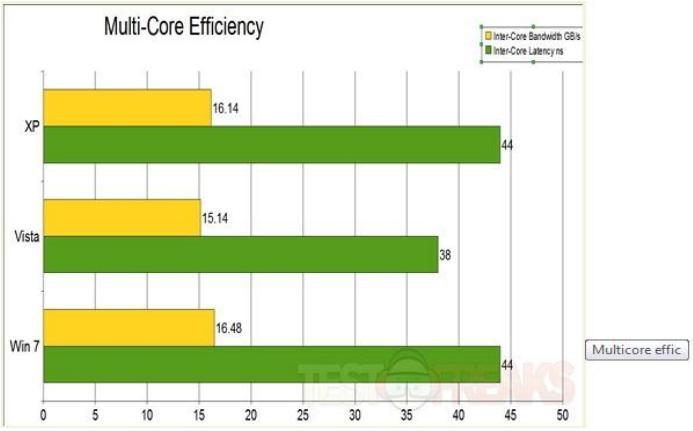
  La principal razón por la que se ha optado por la versión 7 de Windows, con arquitectura de 32bits, ya que las de 64bits presentan bastantes problemas de incompatibilidad con aplicaciones, es por el factor de rendimiento y velocidad). Estudios demuestran que la versión Windows Vista es la menos popular y más lenta,

aparte de consumir una gran cantidad de recursos de la máquina sobre la que se ejecuta, por lo que Windows Vista queda descartado.

Entre las dos opciones restantes, es difícil decidirse, ya que ambas versiones, XP y 7, tienen rendimientos muy similares. Pero en cuando a las capacidades de uno y de otro, es cierto que Windows 7 tiene más servicios y prestaciones, por lo que será la opción elegida.



EVS. Figura 7. Resultados pruebas caché y memoria.



EVS. Figura 8. Resultados multi.-core.

- **Visual Studio 2010:** Se trata de una herramienta que proporciona un entorno de desarrollo integrado único, simplificando la creación y depuración de aplicaciones basadas en MVC. Soporta varios lenguajes y permite a los desarrolladores crear aplicaciones, sitios web y servicios en cualquier entorno que soporte .NET.

Incluye las mejoras que tenía su predecesor Visual Studio 2008, como herramientas para el desarrollo de características de Windows, el Ribbon de WPF y soporte de compilación para F#.

EVS. Figura 9. Visual Studio.

Otra de sus características es su capacidad para desacoplar ventanas de su sitio original y acoplarlas en el lugar más cómodo para el desarrollador.

Se ha elegido el entorno de desarrollo de Visual Studio 2010 porque es la versión más actual, es estable, es robusta y tiene gran popularidad, además de ofrecer múltiples opciones en cuanto a lenguajes, desarrollo de servicios web, etc., además, del gran soporte que le proporciona su creador, Microsoft. Para más información consultar el apartado "*4.4.1. MICROSOFT VISUAL STUDIO* 2008-2010".

- **C#:** Ha sido diseñado específicamente para trabajar sobre la plataforma .NET por tanto va a resultar mucho más sencillo programar usando este lenguaje que usando cualquier otro (que también pueda escribir código en esta plataforma). Por esta razón se dice que C# es el único lenguaje nativo de .NET. Uno de los motivos que impulsaron a Microsoft a la creación de este nuevo lenguaje fue la consolidación de Java como lenguaje de programación dentro del ámbito de Internet, puesto que proporcionaba la flexibilidad, la potencia y la portabilidad que necesitan las aplicaciones web. Por otro lado debido a la gran cantidad de programadores que utilizaban sus anteriores lenguajes, C y C++, debían crear un lenguaje que ofreciese

mecanismos de bajo nivel para que los programadores pudiesen sacar todo el partido de la nueva plataforma .NET.

"Es un lenguaje 70% Java, 10% C++, 5% Visual Basic, 15% nuevo".

Es por esto que C# se creó como un lenguaje orientado a objetos que recoge las características más avanzadas de Java, así como las más potentes de C++. Esto permitirá acceder con C# a todos los servicios que brinde la plataforma .NET, al igual que permitirá crear código muy eficiente en aquellos puntos de la aplicación que sean críticos, así como acceder a las interfaces de programación de las aplicaciones (APIs) existentes. En concreto, C# está especialmente preparado para acceder a la API de Windows y a los objetos COM+ y DLL del sistema, por lo que las aplicaciones escritas en este lenguaje podrán aprovechar todas las características de este popular sistema operativo y de .NET.

En concreto, C# está especialmente preparado para acceder a la API de Windows y a los objetos COM+ y DLL del sistema, por lo que las aplicaciones escritas en este lenguaje podrán aprovechar todas las características de este popular sistema operativo y de .NET. La sintaxis y estructuración de C# es similar a la de C++, ya que Microsoft pretende con esto facilitar la migración de los programadores en C y C++ al nuevo lenguaje. Así mismo C# ofrece también la sencillez y productividad propias de Visual Basic. Con respecto a Java, también hay un cierto parecido sintáctico si se habla de programas sencillos hasta el punto de poder confundirlos. Sin embargo las diferencias en este campo se hacen notables según se profundiza en el lenguaje. Alguno de los parecidos con Java y que no contiene C++ sería el contar con un recolector de basura, manejar excepciones, no soportar herencia múltiple o ser un leguaje orientado a objetos puro. Para información más detalladas acerca de las características de C#, consultar el apartado *"4.3.1. C#"*.

- **Firebird**: Firebird es un sistema gestor de bases de datos relacional. Como tal, está diseñado para soportar la creación y mantenimiento de estructuras de datos

abstractas, no sólo almacenar datos sino también mantener las relaciones y optimizar la velocidad y consistencia cuando los datos pedidos son enviados a los clientes. En su conjunto, los objetos definidos en una base de datos son conocidos como metadatos o esquema. En Firebird se definen los siguientes tipos de objetos básicos:

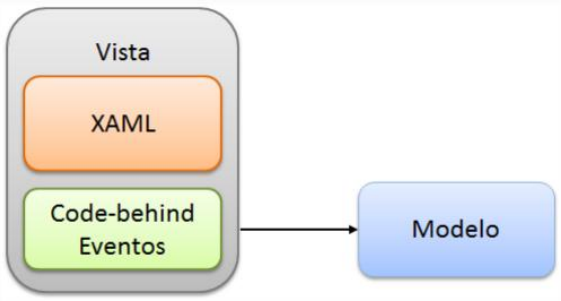
- La propia base de datos.
- Los dominios.
- Las tablas
- Los índices.
- Las vistas.
- Los procedimientos almacenados y los triggers.
- Las excepciones.

Para más información dirigirse al apartado "FIREBIRD, LA ALTERNATIVA A MYSQL", en el Capítulo III.

- **XAML**: Las aplicaciones WPF pueden ser usadas tanto como programas de escritorio o como páginas web. El objetivo de dicho pilar o subsistema es la unificación de todos los elementos gráficos visibles por el usuario tales como tipografías, documentos, textos, dibujos bidimensionales y tridimensionales, luces, escenas, cámaras, vídeos, audios, etc.

  Para otorgar a esta nueva generación de aplicaciones de escritorio y web un marcado carácter técnico, todo el subsistema gráfico explicado y consecuentemente las aplicaciones desarrolladas bajo él, se basan en un nuevo patrón arquitectónico denominado MVVM (Model-View-ViewModel). Para muchos desarrolladores de software, dicho patrón no lo es como tal o bien es un plagio readaptado de Microsoft al patrón arquitectónico. MVVM comparte todos los beneficios de MVC, pero tiene un beneficio adicional: la simplificación que resulta de usar binding declarativo para transferir los datos desde y hacia el modelo a la vista.

Sin embargo, dicho patrón cumple varias funcionalidades muy importantes a la hora de gestionar un proyecto software moderno y adaptado a las necesidades de hoy que ofrecen las interfaces gráficas.



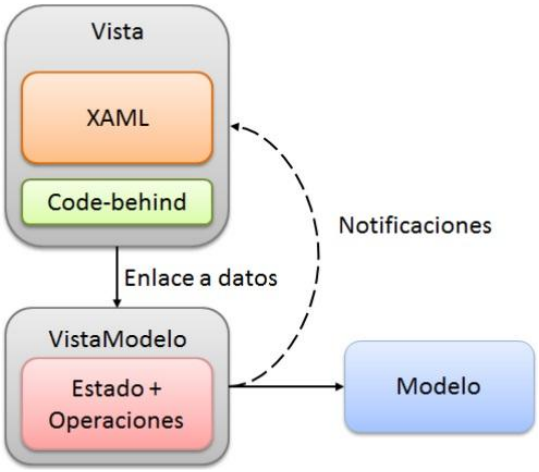
EVS. Figura 10. Modelo MVC.

La ventaja más notable introducida por el patrón MVVM es la siguiente:

- **Vista**: Para desarrollar la interfaz gráfica se requiere de XAML pero no se requiere de código programado, aunque en este proyecto vamos a suprimir todo el código que podamos en C# e implementarlo en XAML. Es importante destacar que XAML define la interfaz gráfica y su comportamiento, pero no la lógica del código (flujos condicionales o bucles).La interfaz gráfica se encuentra separada en ficheros propios a los cuales el equipo de diseñadores puede gestionar. Es importante a la hora de paralelizar el grupo de trabajo y delegar las funciones de diseño a aquellos diseñadores entrenados para dicha tarea.
- **Vista-Modelo**: Se ha creado una interfaz sólida y consistente de comunicaciones entre la capa Vista y la capa Vista-Modelo para poder interactuar con los objetos gráficos sin necesidad de introducir código en la capa Modelo.

  Los controles y las acciones gráficas permanecen en un contexto gráfico donde los diseñadores pueden realizar su trabajo.

- **Modelo**: Se pueden realizar clases para testear las clases del modelo lo que aumenta el grado de robustez de las aplicaciones.

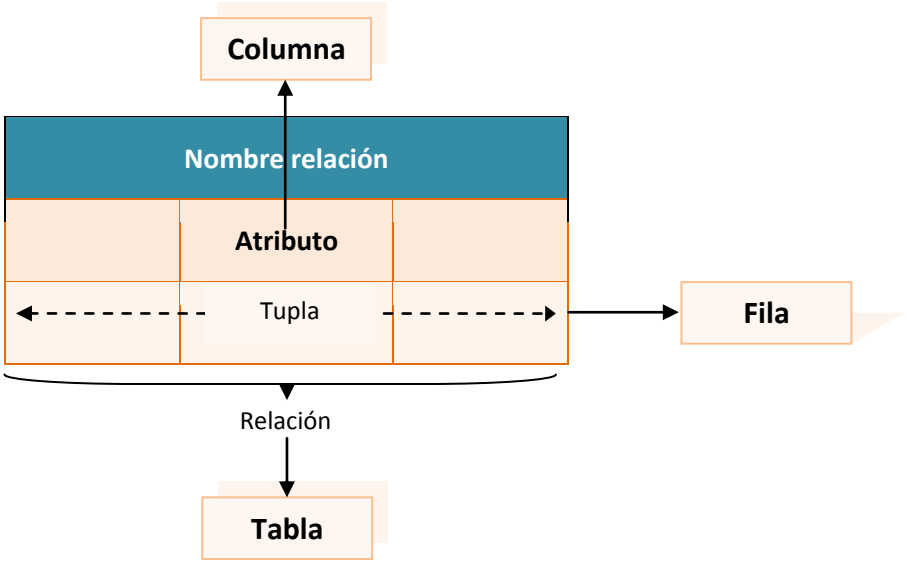


EVS. Figura 11. Modelo MVVC.

- **WPF:** Consiste un subsistema gráfico para el renderizado de interfaces de usuario para aplicaciones basadas en Windows. Este subsistema está dividido en las siguientes capas:

  - **PresentationFramework:** Representa la capa visible del framework de presentación de gráficos.
  - **PresentationCore**: Representa el núcleo o la capa de aceptación de peticiones para ser procesadas.
  - **Common Language Runtime:** Representa el núcleo de .NET por la cual todas las peticiones de la máquina virtual son procesadas.

- **SQL**: Como ya se ha mencionado, se trata de un lenguaje declarativo que sirve para interactuar con modelos relacionales de datos mediante un lenguaje de manipulación de datos. Las instrucciones SQL son descompuestas por el SGBD en operaciones relacionales. La información necesaria para conocer el lenguaje SQL se puede encontrar en el apartado "*2.2.5. LENGUAJE DE CONSULTA ESTRUCTURADO* SQL" del Capítulo II.



EVS. Figura 12. SQL y el modelo relacional de datos

A modo de resumen, podemos enunciar las siguientes características de SQL:

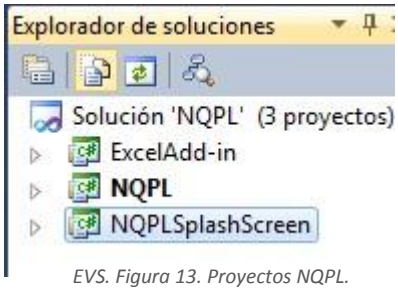
- Es un lenguaje basado en el idioma Inglés que usa frases en inglés para manipular la base de datos.
- Es no procedimental. Se especifica la información requerida, no las operaciones o la forma de llegar a la información. Cada SGBD tiene un optimizador de búsquedas, el cual traduce sus cláusulas en SQL y elabora el camino o la ruta óptima para llegar a los datos.
- Cuando se trabaja con los datos, todas las filas afectadas por los comandos se tratan como un solo bloque, no son tratadas una por una.
- Adicionalmente, SQL permite que los resultados de una consulta sean los datos de entrada para una nueva

- **FIREBIRD**:

7.2. DESCRIPCIÓN DE LOS COMPONENTES DE LA SOLUCIÓN

En el presente apartado se describen los componentes del sistema de traducción de sentencias. La solución NQPL consta de 3 proyectos principales:

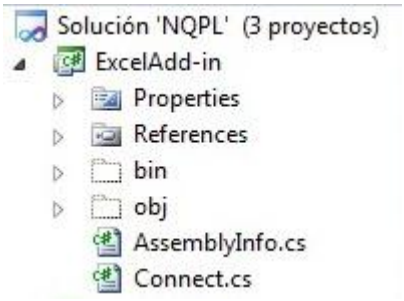
- Proyecto Excel-Addin.
- Proyecto NPQL.
- Proyecto NQPL SplashScreen.



EVS. Figura 13. Proyectos NQPL.

EXCEL-ADDIN

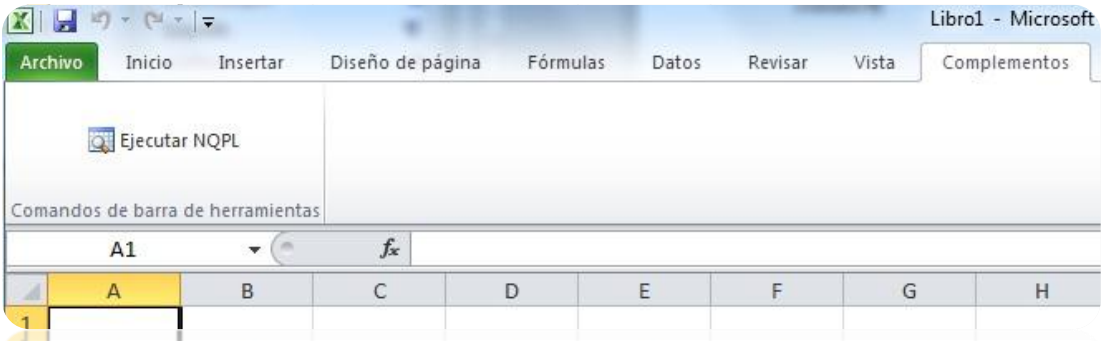
Este proyecto es el encardado de la integración de la solución NQPL como componente COM en el paquete de productividad Microsoft Office.



EVS. Figura 14. Proyecto Excel Add-in.

En este proyecto, para agregar la funcionalidad del registro del Add-in, se ha creado la clase Connect:

Connect: Esta clase es la encargada de todo el proceso de creación de la conexión Interop entre la aplicación y Office. En ella se crea el objeto de office COM que se registrará como componente y se agregará a la barra de herramientas de office. Este objeto COM consiste un *CommandBarButton* que se añade a un *CommandBar* de office mediante VSTO que definirá cuál será la interfaz de la clase *Connect.cs*.

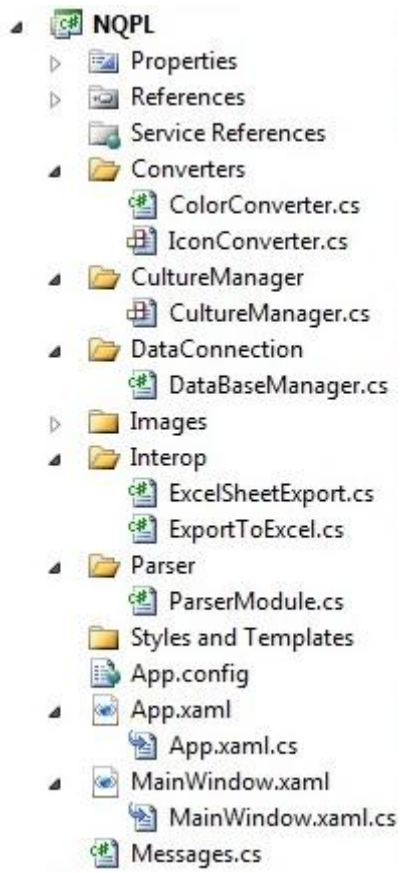


EVS. Figura 15. Integración del Add-in con Microsoft Office.

Para completar la integración, se asigna a ese *CommandBarButton* un evento click que llame a NQPL.exe y ejecute la aplicación. Este componente quedará registrado y activado como componente Office, cargándose y descargándose cada vez que se ejecute la aplicación Office que corresponda.

NQPL

Este proyecto engloba toda la funcionalidad relacionada con la interfaz de traducción de sentencias.



EVs. Figura 16. Componentes de la solución.

ColorConverter.cs: Conversor de color para pasar los formatos *Hex* y *RGB* a *SolidColorBrush*.

IconConverter.cs: Conversor de iconos de .ico a .bmp para los objetos COM de Microsoft Office.

CultureManager.cs: Gestor de los recursos de idioma de la aplicación.

DataBaseManager.cs: Gestor de la conexión del sistema con la base de datos mediante y *FBDatabaseConnector* y el *FBAdapter*. En esta clase tiene lugar el intercambio de datos

entre los elementos del proveedor de datos .NET de Firebird y los objetos ADO.NET de Visual Studio.

El ***FBDatabaseConnection*** es el objeto que realiza la conexión con la base de datos, pasándole como parámetros un string con las propiedades necesarias para establecerla.

```
FbConnection oFBConexion = new FbConnection(sConnectionString);
```

El **FBAdapter** es el objeto que lanza la query contra la base de datos y mediante la instrucción *FBAdapter.Fill(DataTable)* se pasan los datos obtenidos al objeto *DataTable* de ADO.

```
FbDataAdapter oDataAdapter = new FbDataAdapter(sQuery,
oFBConexion);
oDataAdapter.Fill(oDataTable);
```

ExcelSheetExport.cs: Clase encargada, mediante Interop, de migrar el *DataGrid* donde se muestran los resultados de la consulta ejecutada a una hoja de Excel. Si la aplicación Microsoft Excel no está ejecutándose, se abre una nueva instancia y se muestran los resultados formateados. Si se está ejecutando, rellena la hoja actual con dichos datos. Ejemplo de código:

```
      // Create an Excel object
excel = new Microsoft.Office.Interop.Excel.Application();

// Add column headings
            int iCol = 0;
            foreach (DataColumn c in dt.Columns)
            {
                iCol++;
                excel.Cells[1, iCol] = c.ColumnName;
                excel.Cells[1, iCol].Interior.Color =
System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.PapayaWhip);
                                    excel.Cells[1, iCol].Font.Color =
 System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Black);
                excel.Cells[1, iCol].Font.Size = 14;

            }
```

```
// for each row of data...
int iRow = 0;
foreach (DataRow r in dt.Rows)
{
    iRow++;

    // Add each row's cell data...
    iCol = 0;
    foreach (DataColumn c in dt.Columns)
    {
        iCol++;
        excel.Cells[iRow + 1, iCol] = r[c.ColumnName];
    }
}
```

ExportToExcel.c: Clase encargada de migrar los datos obtenidos en el *DataGrid* a un archivo Microsoft Excel. Básicamente hace lo mismo que ExcelSheetExport.cs pero en lugar de mostrar la hoja de Excel, muestra una ventana de diálogo que ofrece la posibilidad de seleccionar dónde guardar ese archivo.

ParseModule.cs: Clase encargada del proceso de traducción de las sentencias en Lenguaje Natural a sentencias en lenguaje SQL. En esta clase lo que se hace es analizar el tipo de sentencia que entra y ver si se trata de un Select, un Insert, Delete, etc.

Una vez analizado se procede a la descomposición gramatical por comando Where, si lo hubiere, Operador (=,>,<,igual, menor, mayor, etc.), parte derecha del operador y parte izquierda, limpieza de palabras sin contenido necesario para la traducción (la, lo, el, a, verbos, etc.).Durante el proceso de detección de condición Where, se analiza si hay un carácter de final de línea (o commit) para descartar condiciones múltiples tras el Where. Si las hubiere, se almacenarían en una lista de condiciones y se analizaría cada fragmento, obteniendo su operador, su parte izquierda y su parte derecha:

WHERE OperadorIzquierdo <Operador> OperadorDerecho AND

OperadorIzquierdo <Operador> OperadorDerecho And… “commit”

Una vez limpiada la frase y quedando únicamente palabras clave, se analiza cual es la que define sobre qué tabla se van a realizar las operaciones y el conjunto de columnas

que se quieren mostrar en la operación y que se almacenan en una lista. Una vez obtenidos todos los elementos necesarios para la construcción de la sentencia a traducir se llama al DataBaseManager y se crea una FbConnection que se encargará de obtener el resultado de la base de datos mediante el FBAdapter. A continuación se muestra la línea que compone la sentencia de parseo:

```
oParseResult = Action + " " + sDistinct + ColumsString.ToUpper() + " "
+FromOrTable + " " + TableName + " " + sWhere + " " + LeftComparator +
" " + LeftSideOperator + " " + RightSideOperator.ToString() +
sConditions; ;
```

App.cs: Clase de entrada de ejecución a la aplicación.

MainWindow.xaml: Archivo XAML que contiene la estructura visual (capa de Vista) de la aplicación. En ella se recogen los estilos, templates empleados y el código XAML que WPF interpreta y renderiza. Ejmeplo de código contenido en la clase:

```
<MenuItem Name="MenuFile" Header="_Archivo" Margin="0,4,0,0"
InputGestureText="Alt+a">
                <MenuItem  Name="FileOpen" Header="Abrir Base de Datos"
HorizontalAlignment="Stretch" Background="Transparent"
BorderThickness="0" Click="FileOpen_Click" StaysOpenOnClick="True">
                </MenuItem>
                <MenuItem Name="FileExport" IsEnabled="false"
Header="Exportar a Excel" HorizontalAlignment="Stretch"
Background="Transparent" BorderThickness="0" >
                    <MenuItem Name="ExportToFile" Header="Exportar a
archivo" Click="FileExport_Click"></MenuItem>
                    <MenuItem Name="ExportToSheet" Header="Exportar a
Hoja de Excel" Click="ExportToSheet_Click"></MenuItem>
                </MenuItem>
                <MenuItem Name="FileExit" Header="_Salir"
HorizontalAlignment="Stretch" Background="Transparent"
BorderThickness="0" Click="FileExit_Click"/>
            </MenuItem>
```

MainWindow.xaml.cs: Clase encargada de gestionar los eventos que se producen en la ventana de aplicación y de controlar las llamadas al resto de clases.

Messages.cs: Clase encargada de la instanciación de los distintos tipos de ventanas emergentes que se han incluido en la interfaz.

NQPL SPLASHSCREEN

Program.cs: Entrada del proyecto del SplashScreen.

SplashScreen.cs: Clase que contiene el código necesario para instanciar una SplashScreen En esta clase además se definen parámetros del diseño, duración etc.

7.3. APROBACIÓN DE LA SOLUCIÓN

El proyecto ha sido aprobado por el Jefe de Proyecto, durante una reunión con el comité de dirección. La aprobación ha sido formal, se ha aceptado el presupuesto económico necesario, se ha asegurado el cumplimiento de todos los requisitos y se ha establecido una fecha de fin de ejecución en la planificación.

8. DEFINICIONES Y ACRÓNIMOS

8.1. DEFINICIONES

32bits /64bits Tipo de arquitectura de procesador. El número representa el número de direcciones de memoria que puede mapear, en potencia de 2.

Booch Metodología de desarrollo de software orientado a objetos.

Commit Hace alusión al carácter de cierre de sentencias de interacción con bases de datos, concretamente en SQL.

Core Núcleo del procesador.

Corner Esquina.

Distro Nombre que se le da coloquialmente a las distribuciones de Linux.

F# Lenguaje de programación multiparadigma, para la plataforma .NET, que conjunta la programación funcional con las disciplinas imperativa y orientada a objetos.

Gygabyte Unidad de medida de cantidad de datos. Equivalente a 1024 Megabytes.

Herramienta de Ventana Abstracta Conjunto de herramientas de gráficos, interfaz gráfico y ventanas de Java.

HEX Sistema de numeración en base 16.

Hibernate Herramienta de mapeo relacional entre objetos Java.

Inside-corner Esquina interior (Rincón).

JQuery Biblioteca de JavaScript que permite interactuar con HTML, AJAX, manejar eventos, etc. Es un software libre.

Lenguaje de Modelado Unificado Lenguaje de modelado de sistemas software más popular. Sirve para especificar y describir métodos o procesos.

Megabyte Unidad de medida de cantidad de datos informáticos. Equivalente a 1024 bytes.

Memoria de acceso aleatorio Dispositivo hardware donde el ordenador recibe las operaciones y guarda los resultados.

Model-View-ViewModel Modelo-Vista-VistaModelo. Sucesor del modelo Modelo-Vista-Controlador. Se emplea sobre todo en desarrollos con WPF y Silverlight.

Outside-corner Esquina exterior (Esquina).

Proceso Racional Unificado Metodología de desarrollo de software que junto con UML constituye es el estándar más utilizado en ciclos orientados a objetos.

Revoluciones por minuto Unidad estándar de frecuencia que mide la velocidad angular de un dispositivo sobre su eje.

Scrum Sistema basado en el desarrollo ágil de software enfocado a la gestión de procesos de desarrollo de software. Es un lenguaje gráfico de visualización, especificación de métodos o procesos, construcción y documentación de software.

SplashScreen Imagen que aparece mientras un programa está cargando. Su objetivo es que el usuario sepa que la aplicación está cargando.

Stakeholder Público interesado. Puede ser un individuo o un grupo.

Swing Biblioteca gráfica de Java.

Ultimate Edition Sufijo empleado por Microsoft en sus productos Visual Studio y Windows (a partir de Vista) más completos.

Variable booleana Variable que puede tomar únicamente dos valores: verdadero o falso.

Variable entera Variable que puede tomar valores comprendidos entre -2.147.483.648 y 2.147.483.647.

Velocidad angular Medida de la velocidad de rotación de un dispositivo sólido.

Windows Server 2003, 2008, Windows ME/2000/XP/Vista/7 Sistemas operativos de Microsoft.

8.2. ACRÓNIMOS

AWT Abstract Window Toolkit (Herramienta de Ventana Abstracta).

GB Gygabyte.

IEEE Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

ME Millenium Edition (Edicion Milenio).

MB Megabytes.

MUI Multilingual User Interface (Interfaz de Usuario Multilenguaje).

MVC Modelo-Vista-Controlador.

RAM Randon Access Memory (Memoria de Acceso Aleatorio).

RGB Red Green Blue (Rojo, Verde Azul).

RPM Revolutions Per Minute (Revoluciones por Minuto).

RUP Rational Unified Process. (Proceso Racional Unificado).

UC3M Universidad Carlos III de Madrid.

UML Unified Modeling Language (Lenguaje de Modelado Unificado).

Wine WINdows Emulator (Emulador de WIndows).

W3C O WWWC, World Wide Web Consortium (Consorcio World Wide Web).

9. REFERENCIAS

[1] *Ministerio de Política Territorial y Administración Pública, "Métrica Versión 3. Guía de Referencia", 2001. (http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P60085901274201580632&langPae=es). Último acceso 06-06-2011.*

[2] *Estudio de Viabilidad del Sistema. (http://administracionelectronica.gob.es/recursos/pae_000001029.pdf). Último acceso 14-05-2010.*

[3] *WPF MVVM. (http://msdn.microsoft.com/es-es/magazine/dd419663.aspx). Último acceso 28-06-2011.*

[4] *Hellen Borrie. "The Firebird Book: A Reference for Database Developers".*

[5] *Blog de Oskar Álvarez. WPF DataBinding. Overview. (http://geeks.ms/blogs/oalvarez/archive/2009/07/13/data-binding-en-wpf-1-2.aspx y 2-2). Último acceso 05-06-2011.*

[6] *Blog de Fernando Machado Piriz. (http://fernandomachadopiriz.com/2010/06/09/una-simple-introduccin-al-patrn-model-view-viewmodel-para-construir-aplicaciones-silverlight-y-windows-presentation-foundation/). Último acceso 07-06-2011.*

[7] *LINQ Expressions. (http://msdn.microsoft.com/en-us/library/bb397676(v=VS.90).aspx). Último acceso 27-06-2011.*

[8] *Ribbon. (http://msdn.microsoft.com/en-us/library/ff799534.aspx). Último acceso 15-06-2011.*

[9] *SQ-HAL. (http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura/user_manual.htm). Último acceso 03-06-2011.*

[10] *MSEQ, English Query.*

(http://msdn.microsoft.com/en-us/library/aa198281(v=sql.80).aspx). Último acceso 03-06-2011.

[11] VSTO. "VSTO 3.0 for Office 2007 Programming". (http://www.maartenvanstam.nl/vsto-3-0-for-office-2007-programming-sample-chapter-5-microsoft-office-outlook-programming.pdf). Último acceso 17-06-2011.

[12] Jonás A Montilva C., Nelson Arape y Juan Andrés Colmenares. "Desarrollo de Software Basado en Componentes". (http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03%20Desarrollo%20de%20componentes.pdf). Último acceso 26-05-2011.

[13] R.A.E. (http://www.rae.es/rae.html). Último acceso 28-06-2011.

[14] Wikipedia. (http://es.wikipedia.org/wiki/Wikipedia:Portada). Último acceso 28-06-2011.

APÉNDICE B

ANÁLISIS DEL SISTEMA DE INFORMACIÓN

RESUMEN

Este apéndice contiene el Análisis del Sistema de Información asociado a este proyecto.

A lo largo de la fase de análisis, se va a desarrollar el ASI cuya finalidad es obtener una especificación detallada del sistema de información que debe satisfacer las necesidades de los usuarios y que constituya los pilares fundamentales para el diseño de SQPL. Más información en el apartado *"4.2.5. DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI)"*.

HOJA DE ESTADO DEL DOCUMENTO

NQPL		
ASI.HOJA DE ESTADO DEL DOCUMENTO		
Versión	**Fecha**	**Motivación**
0.01	15-06-2011	Primera versión

ASI.Tabla 1. Formato hoja de estado del documento.

REGISTRO DE CAMBIOS

NQPL.ANÁLISIS DEL SISTEMA DE INFORMACIÓN		
ASI.HISTORIAL DE CAMBIOS		
Versión	**Fecha**	**Motivación**
0.01	15-06-2011	Primera versión

ASI.Tabla 2. Formato historial de cambios.

NQPL			
ASI.VALIDACIÓN DEL DOCUMENTO			
Versión	**Fecha**	**Cargo**	**Nombre y Apellidos**
0.01	15-06-2011	Jefe de Proyecto	Carlos Cócera Pérez

ASI.Tabla 3. Formato validación del documento.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. DESCRIPCIÓN Y OBJETIVOS

El Análisis del Sistema de Información tiene como objetivo la especificación detallada del sistema de la información suficiente para que satisfaga las necesidades de los usuarios y constituya el pilar fundamental sobre el que se asiente el diseño del sistema.

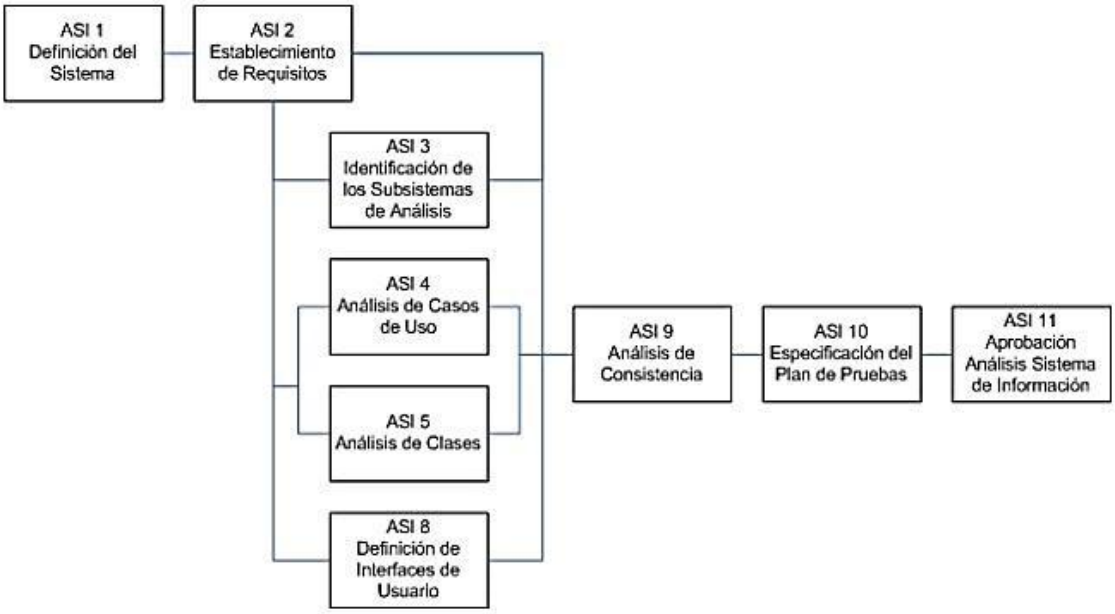
En este documento se determinarán las necesidades de usuario que se han de resolver, se marcarán las pautas necesarias a seguir para conseguir alcanzar objetivos, se definirán la estructura y funcionamiento del sistema y evaluarán los pasos previos al diseño de un sistema nuevo.

Esto será posible gracias a la composición de los siguientes documentos:

- Definición del Sistema.
- Catálogo de requisitos.
- Identificación de subsistemas de análisis.
- Análisis de casos de uso.
- Elaboración del modelo de datos.
- Elaboración del modelo de procesos.
- Definición de interfaces de usuario.
- Análisis de consistencia y especificación.
- Aprobación del ASI.

Como se ha mencionado, lo que se persigue con estas tareas es definir el problema que se va a resolver, no resolverlo. En esta fase de análisis se descompone el problema o problemas principales en problemas más pequeños con el fin de facilitar la tarea de su análisis y simplificar la respuesta a ellos.

En conclusión, el desarrollo del documento ASI pretende delimitar el ámbito que cubrirá el sistema, la arquitectura de una sesión de consultas con NQPL y la arquitectura lógica para el software final. En la siguiente figura, se aprecia la secuencia de tareas que el desarrollo del ASI implica:



ASI.Figura 1. Secuencia de tareas en la fase del ASI.

2. DEFINICIÓN DEL SISTEMA

Esta actividad tiene como objetivo efectuar una descripción del sistema, delimitando su alcance, estableciendo las interfaces con toros sistemas e identificando a los usuarios. En el caso de tareas que se hayan realizado durante la fase del EVS, apéndice A se tomarán estas como punto de partida.

2.1. DETERMINACIÓN DEL ALCANCE DEL SISTEMA

En términos generales, la aplicación que se va a implementar se basa en el análisis, diseño y construcción de un componente integrado que permita un diálogo natural entre un sistema artificial de información basado en lenguaje SQL y una persona. El sistema estará conectado a una base de datos concreta y definida por cliente.

Este proyecto no deriva ni es continuación de ninguno previo, sino que se ha iniciado como respuesta a la necesidad de este tipo de subsistemas en el ámbito tecnológico de hoy en día. La funcionalidad completa se definirá detalladamente a lo largo del presente documento y los requisitos establecidos se aplicarán a NQPL una vez aprobados por el cliente y por el ASI.

2.2. MODELO DE DOMINIO

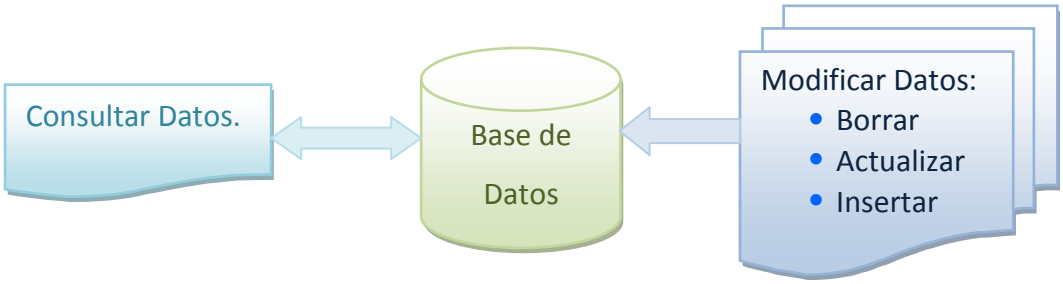
Como se ha mencionado en el apartado anterior, el dominio al que nos enfrentamos se trata de un sistema gestor de bases de datos. El sistema diseñado se conectará a una base de datos que ya está implementada y definida previamente y permitirá al usuario realizar operaciones sobre ella mediante un lenguaje coloquial.

Habrá un único tipo de Usuario. No es necesario establecer una diferenciación de usuarios, ya que es un sistema que está diseñado para que cualquiera pueda, no sólo realizar consultas a la base de datos, sino que podrán modificar la información, sin que exista necesidad de que haya un administrador. La aplicación deberá ser lo suficientemente sencilla

como para permitir esto, es decir, habrá un control total sobre la misma sin restricciones funcionales.

2.3. MODELO DE NEGOCIO

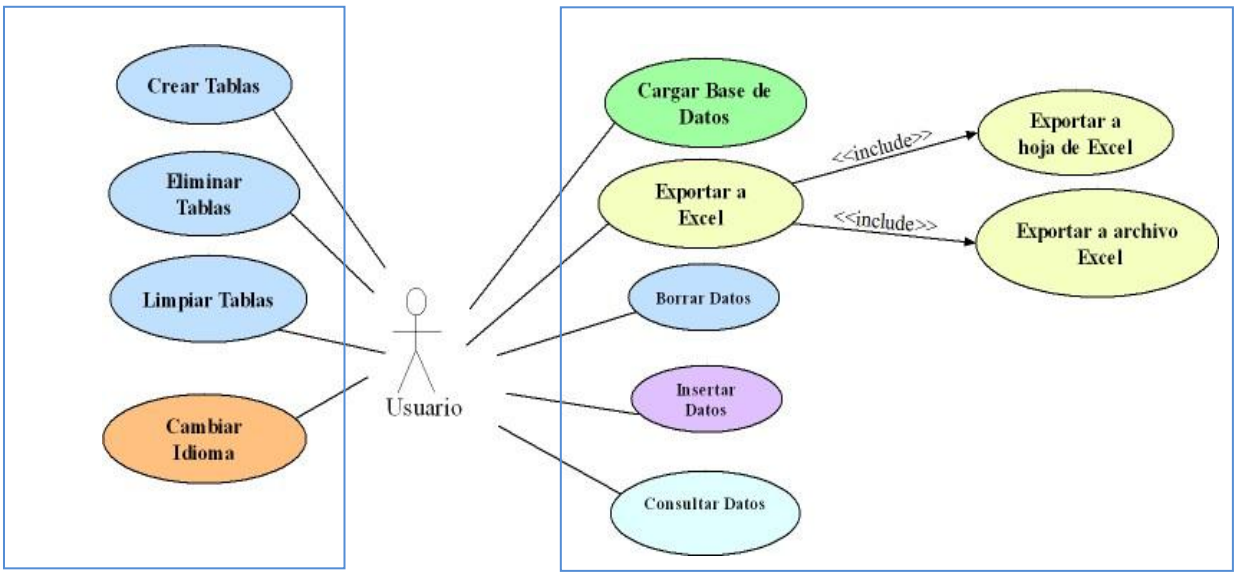
El alcance del sistema queda delimitado por las operaciones que podrá realizar un usuario con los elementos del dominio.



ASI.Figura 2. Ciclo de tareas en la fase del ASI.

2.4. USUARIOS DEL SISTEMA

A continuación se muestra el diagrama de casos de uso que muestra el conjunto de operaciones que se pueden realizar desde el sistema sobre la base de datos. Se entrará en una especificación de casos de uso más detallada en el apartado "*4. MODELO DE CASOS DE USO*



ASI.Figura 3. Diagrama de Casos de Uso.

4. ”.

2.5. PLAN DE TRABAJO

Para esta actividad se han llevado a cabo 4 tareas:

ACTIVIDAD	DESCRIPCIÓN
Primera	Análisis de la situación actual en el ámbito de investigación y desarrollo de las ILNBD.
Segunda	Análisis de sistemas similares de traducción de lenguaje natural a SQL. Derivan de este estudio posibles funcionalidades que agregar o mejorar en el proyecto NQPL.
Tercera	Identificación de los requisitos funcionales y no funcionales del sistema. A partir de estos, se definirán los siguientes: • Elaboración del diagrama E/R. • Elaboración del modelo de negocio. • Elaboración del diagrama de casos de uso del sistema.
Cuarta	A partir de la actividad tercera se procede al diseño del sistema.

ASI.Tabla 4. Actividades del plan de trabajo.

3. REQUISITOS DE SOFTWARE

Esta actividad incluye la determinación de los requisitos de software generales. Una vez finalizadas, se analiza la información obtenida definiendo los requisitos y sus prioridades.

3.1. ESPECIFICACIÓN DE REQUISITOS

Los requisitos tendrán una plantilla de presentación, a modo de tabla, que estará estandarizada de acuerdo al estándar PSS-05-02 adaptado y que recogerá los diferentes atributos de cada uno de ellos. Los distintos atributos, su significado y los valores que podrán admitir se regirán por las siguientes reglas:

Identificador: Sirve para referenciar a cada requisito unívocamente. Obedecen a la siguiente regla de codificación, donde RS indica que se trata de un requisito de software:

RS-XXX-ZZ<Numeración>

XX se corresponderá con dos caracteres alfanuméricos y harán referencia al tipo de requisito, pudiendo tomar los siguientes valores:

FUN Requisitos funcionales.

NFN Requisitos no funcionales: El sufijo ZZ identificará a subgrupo de requisitos no funcionales Dentro de los no funcionales se pueden distinguir las siguientes categorías:

- CA Requisito de calidad.
- CO Requisito de comprobación.
- DC Requisito de documentación.
- FU Requisito funcional.
- IN Requisito de interfaz.
- MN Requisito de mantenimiento.

- OP Requisito operacional.
- PO Requisito de portabilidad.
- RC Requisito de recursos.
- RN Requisito de rendimiento.
- SG Requisito de seguridad.
- RU Requisito de usabilidad.
- VR Requisito de verificación.

Referencias: Contiene los identificadores de los requisitos de software que estén relacionados con él.

Descripción: Se deberá dar una breve descripción del requisito.

Prioridad: Grado de importancia y urgencia en la implementación del requisito. Los posibles valores que puede tomar son: *Alta, Media o Baja.*

Estabilidad: Mide el grado en el que el requisito es susceptible a cambiar, o no, a lo largo de la vida del proyecto. Puede tomar los valores *Alta, Media o Baja.*

Fuente: Indica la procedencia del requisito y tiene como objetivo dar información de cuál es la fuente para poder acceder a una información acerca del mismo más amplia. Los valores que puede tomar son:

- Desarrollo*:* El requisito ha sido puesto por el equipo de desarrollo, más en concreto por el Analista Funcional.
- Cliente*:* El requisito ha sido puesto por el equipo de UC3M.

Claridad: Determina la falta de ambigüedad del requisito. Puede tomar los valores *Alta, Media o Baja.*

Verificabilidad: Mide cuán fácil se puede comprobar la incorporación del requisito en el sistema. Puede tomar los valores *Alta, Media o Baja.*

Los sucesivos apartados se dedicarán a la especificación de los requisitos de software identificados y catalogados según su tipo.

3.2. REQUISITOS FUNCIONALES

En este apartado se especificarán con detalle aquellos requisitos del sistema que tienen que ver con la funcionalidad. Puede existir algún requisito que previamente haya sido identificado en el apartado "*4. DEFINICIÓN DE REQUISITOS DEL* SISTEMA".

RS-FUN-001

Referencias: RUR-007.

Descripción: La invocación del sistema se realizará desde un componente Office.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-002

Referencias: RUR-001.

Descripción: El sistema deberá poder cargar una base de datos SQL.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-003

Referencias: RUR-002, RUR-004, RUC-001, RUC-002. RUC-004.

Descripción: El sistema deberá permitir insertar datos.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-004

Referencias: RUC-001, RUC-002, RUC-011.

Descripción: El sistema deberá permitir borrar Tablas (Drop).

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-005

Referencias: RUC-001, RUC-002, RUC-003 RUC-006.

Descripción: El sistema generará una sentencia SQL en lenguaje formal.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-006

Referencias: RUR-001, RUC-002, RUC-003.

Descripción: El sistema permitirá modificar la sentencia SQL generada y ejecutarla.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-007

Referencias: RUC-012, RUC-013.

Descripción: El sistema permitirá exportar el resultado aun fichero Excel.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-008

Referencias: RUC-015.

Descripción: El sistema permitirá borrar tablas (Truncate).

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-FUN-009

Referencias: RUR-002, RUC-002, RUC-007.

Descripción: El sistema permitirá borrar datos (Delete).

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-010

Referencias: RUC-005, RUC-009.

Descripción: El sistema permitirá al usuario cambiar de idioma.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-011

Referencias: RUC-19.

Descripción: El sistema permitirá insertar el texto de los campos de una tabla mediante el visor de tablas del sistema haciendo clic derecho sobre la columna o el campo.

Prioridad: Media**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-012

Referencias: RUC-001, RUC-002, RUC-003.

Descripción: El sistema deberá permitir ejecutar sentencias SQL directamente sin previamente traducirlas del lenguaje natural.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-013

Referencias: RUC-19.

Descripción: El sistema permitirá introducir campos o nombres de columnas en la caja de texto de lenguaje SQL.

Prioridad: Baja**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-14

Referencias: RUC-16, RUC-18.

Descripción: El sistema permitirá exportar el resultado de una consulta a una hoja Excel.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-015

Referencias: RUC-17.

Descripción: El sistema deberá permitir lanzar la aplicación desde un componente Office (Excel, Word..).

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-FUN-016

Referencias: RUC-001, RUC-002, RUC-003, RUR-015, RUR-16, RUR-17, RUR-18, RUR-19.

Descripción: El sistema deberá permitir realizar consultas (Select).

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

3.3. REQUISITOS NO FUNCIONALES

Dentro de los requisitos no funcionales se han desglosado distintas categorías para diferenciar unos de otros y que no exista posibilidad de equívocos. Estos requisitos no describen ni información a guardar ni funciones a realizar, es decir, no alteran el estado del sistema.

REQUISITOS DE CALIDAD

RS-NFN-CA001

Referencias: RUR-005

Descripción: Se cargará tanta funcionalidad de C# como sea posible en XAML.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-NFN-CA002

Referencias: RUR-010.

Descripción: Las representaciones en XAML seguirán recomendaciones del estándar definido por la W3C y se ceñirá a los GUI Guidelines publicados por Microsoft en MSDN.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta.

REQUISITOS DE COMPROBACIÓN

RS-NFN-CO001

Referencias:

Descripción: La aplicación permitirá comprobar al consulta traducida antes de ejecutarla para evitar lanzar sentencias incorrectas.

Prioridad: Media**.**

Estabilidad: Media**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta.

REQUISITOS DE DOCUMENTACIÓN

RS-NFN-DC001

Referencias:

Descripción: Todo el proceso de codificación seguirá el estándar definido en los Coding Guidelines.

Prioridad: Media**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Media.

RS-NFN-DC002

Referencias: -

Descripción: El proceso de implementación del sistema deberá ir en la línea marcada por los estándares de diseño seleccionados.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta.

RS-NFN-DC003

Referencias: -

Descripción: Se documentará cualquier tipo de cambio o implementación relevante en el sistema que sea de interés para los desarrolladores, tanto los que actualmente forman parte del equipo como para las nuevas incorporaciones.

Prioridad: Media**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Media.

REQUISITOS DE INTERFAZ

RS-NFN-IN001

Referencias: RUC-001, RUC-006.

Descripción: La aplicación deberá mostrar el resultado de las operaciones de consulta en la ventana de aplicación.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN002

Referencias: RUC-002

Descripción: El sistema deberá permitir cargar el contenido de una tabla de la base de datos cargada en un visor de campos.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN003

Referencias: -

Descripción: El sistema deberá permitir navegar por el sistema de bases de datos y de tablas en un árbol de jerarquía.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN004

Referencias: RUR-010.

Descripción: Las representaciones en XAML seguirán recomendaciones del estándar definido por la W3C y se ceñirá al estándar de diseño de Interfaces publicada por Microsoft en MSDN.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-IN005

Referencias: -

Descripción: El sistema deberá permitir introducir operaciones en Lenguaje Natural a través de la aplicación en una caja de texto específica para esta tarea.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-IN006

Referencias: -

Descripción: El sistema deberá permitir introducir operaciones de bases de datos en lenguaje SQL a través de la aplicación en una caja de texto específica para esta tarea.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN007

Referencias: -

Descripción: El sistema permitirá ocultar y mostrar mediante un botón el visor de campos de la tabla para aumentar el espacio de trabajo al usuario.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN008

Referencias: -

Descripción: El sistema permitirá ocultar y mostrar mediante un botón el visor de campos de la tabla para aumentar el espacio de trabajo al usuario.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN009

Referencias: -.

Descripción: Se podrá maximizar, minimizar o normalizar la ventana de aplicación.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN010

Referencias: -

Descripción: Se podrá modificar el tamaño de los elementos de internos de la ventana de aplicación.

Prioridad: Media**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN011

Referencias: -

Descripción: Se podrá modificar el tamaño de los elementos de internos de la ventana de aplicación.

Prioridad: Media**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN012

Referencias: -

Descripción: La aplicación no permitirá exportar datos si no se ha realizado correctamente una operación y se ha obtenido un resultado factible para la exportación.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN013

Referencias: -

Descripción: La aplicación inactivará las opciones de exportación de datos si no hay un resultado obtenido en la consulta.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-IN014

Referencias: -

Descripción: La aplicación activará las opciones de exportación de datos si hay un resultado obtenido en la consulta.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-IN015

Referencias: RUC-020.

Descripción: La aplicación deberá migrar los datos de la consulta obtenida en Excel debidamente formateados.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

REQUISITOS DE MANTENIMIENTO

(No aplicable).

REQUISITOS OPERACIONALES

RS-NFN-OP001

Referencias: -.

Descripción: El usuario deberá ser totalmente ajeno al proceso de traducción.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-OP002

Referencias: RUR-011, RUR-012.

Descripción: La aplicación estará destinada a cualquier tipo de usuario.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-OP003

Referencias: RUR-015.

Descripción: Las sentencias en lenguaje natural deberán finalizar con “.”.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Desarrollo.

Claridad: Alta.

Verificabilidad: Alta**.**

RS-NFN-OP004

Referencias: RUR-016.

Descripción: Las sentencias con condiciones múltiples deberán separarse con la palabra “además”. Esto se debe a que separarlas con “y” únicamente no es factible para el procesado.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-OP005

Referencias: RUR-017.

Descripción: En operadores, si es el derecho y es más de una cadena, se pondrá entrecomillado. Por ejemplo, X = ‘Y ZZ TTT’.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-OP006

Referencias: RUR-018.

Descripción: No se podrán hacer Select anidados.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

REQUISITOS DE RECURSOS

RS-NFN-RC001

Referencias: RUR-007.

Descripción: La ejecución de la aplicación deberá tener instalado Microsoft Office 2007 o superior.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Cliente**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

RS-NFN-RC002

Referencias: RUR-002 RUR-007.

Descripción: El entorno de ejecución deberá tener instalados Windows XP o superior y .NET 3.X.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-RC003

Referencias: RUR-002 RUR-007.

Descripción: El entorno de ejecución deberá tener las características de hardware mínimas establecidas para ejecutar la aplicación.

Prioridad: Media.

Estabilidad: Alta.

Fuente: Cliente.

Claridad: Alta.

Verificabilidad: Alta.

RS-NFN-RC004

Referencias: -.

Descripción: Hay que tener instalado el proveedor de servicios de Firebird.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta**.**

REQUISITOS DE PORTABILIDAD

(No aplicable).

REQUISITOS DE RENDIMIENTO

(No aplicable).

REQUISITOS DE SEGURIDAD

RS-NFN-SG01

Referencias: RUR-012.

Descripción: El sistema de ficheros deberá estar controlado por Sourcesafe en su repositorio correspondiente.

Prioridad: Alta**.**

Estabilidad: Alta**.**

Fuente: Desarrollo**.**

Claridad: Alta**.**

Verificabilidad: Alta.

REQUISITOS DE VERIFICACIÓN

(No aplicable).

4. MODELO DE CASOS DE USO

4.1. DESCRIPCIÓN

La tarea que se desarrolla en este apartado se basa en la descripción de elementos o usuarios externos al sistema (actores) y de la funcionalidad del sistema (casos de uso).

Un Modelo de Casos de Uso describe los requerimientos funcionales de un actor (usuario, sistema, dispositivo, etc.) en términos de las interacciones que éste ejecuta con el sistema. El modelado de casos de uso es una técnica efectiva y a la vez simple para modelar los requerimientos del sistema desde la perspectiva del usuario. Presenta el sistema desde la perspectiva de su uso y esquematiza como proporcionará valor a sus usuarios.

El Modelo de Casos de Uso sirve como acuerdo entre clientes y desarrolladores para limitar las funciones con que dispondrá el sistema luego de ser implementado, además proporciona la entrada fundamental para el análisis, el diseño, la implementación y las pruebas. Para representar los diagramas del Modelo de Casos de Uso se puede emplear el diagrama de UML de Caso de Uso que sigue la siguiente plantilla:

4.2. DIAGRAMA DE CASOS DE USO

Se ha definido un único escenario, el del usuario/cliente de la aplicación, ya que no se han reconocido otros actores que pudieran interactuar con el sistema y cuyo perfil pueda diferenciarse del identificado. Para la definición de los casos de uso se seguirá la plantilla que se muestra a continuación:

CU-XXX	
Nombre:	
Actores	
Objetivo	
Precondiciones	
Poscondiciones	
Escenario básico	
Escenario alternativo	

ASI.Tabla 5. Plantilla casos de uso.

Cada uno de los atributos recogidos en la tabla se corresponde con la siguiente definición:

Identificador: Cada caso de uso incluirá un identificador que será único para cada uno de ellos. Se nombran con las siglas CU (caso de uso) seguidos de un guión y tres dígitos que enumerarán secuencialmente a los casos de uso.

Nombre: Indica el nombre del caso de uso.

Actores: Participantes en el caso de uso. El único actor identificado será U y representará a cualquier tipo de usuario de la aplicación.

Objetivo: Define cuál es el objetivo del caso de uso.

Precondiciones: Define qué condiciones deben producirse para que el caso de uso pueda tener lugar.

Poscondiciones: Define qué condiciones pueden producirse una vez que el caso de uso ha tenido lugar.

Escenario básico: Pasos para la ejecución del caso de uso.

Escenario alternativo: Posibles caminos alternativos a la ejecución del caso de uso.

CU-001	
Nombre:	Cargar Base de datos en el sistema.
Actores	U
Objetivo	Seleccionar una base de datos almacenada y cargarla en la aplicación para mostrarla.
Precondiciones	-
Poscondiciones	-
Escenario básico	1. 1. Se pulsa la opción Archivo/Abrir Base de Datos 2. Se selecciona una base de datos del sistema. 3. Se pulsa el botón Ok.
Escenario alternativo	1. 2. Se pulsa la combinación de teclas ALT+O 2. Se selecciona una base de datos del sistema. 3. Se pulsa el botón OK.

CU-002	
Nombre:	Realizar una consulta a la base de datos (Select).
Actores	U
Objetivo	Realizar una consulta a la base de datos para que esta muestre un resultado.
Precondiciones	CU-001
Poscondiciones	-
Escenario básico	1. 1. Se introduce la consulta en lenguaje natural. 2. Se pulsa el botón “Traducir Sentencia”. 3. Se pulsa el botón “Ejecutar Sentencia”.
Escenario alternativo	1. 2. Se introduce la consulta en lenguaje SQL. 2. Se pulsa el botón “Ejecutar Sentencia”.

CU-003	
Nombre:	Realizar una operación de inserción en la base de datos (Insert)
Actores	U
Objetivo	Se introduce una sentencia del tipo Insert en la base de datos.
Precondiciones	CU-001
Poscondiciones	-
Escenario básico	1. 1. Se introduce la sentencia Insert en lenguaje natural. 2. Se pulsa el botón “Traducir Sentencia”. 3. Se pulsa el botón “Ejecutar Sentencia”.
Escenario alternativo	1. 2. Se introduce la entencia Insert en lenguaje SQL. 2. Se pulsa el botón “Ejecutar Sentencia”.

CU-004	
Nombre:	Realizar una operación de borrado en la base de datos (Delete).
Actores	U
Objetivo	Se introduce una sentencia del tipo Delete en la base de datos.
Precondiciones	CU-001
Poscondiciones	-
Escenario básico	1. 1. Se introduce la sentencia Delete en lenguaje natural. 2. Se pulsa el botón “Traducir Sentencia”. 3. Se pulsa el botón “Ejecutar Sentencia”.
Escenario alternativo	1. 2. Se introduce la sentencia Delete en lenguaje SQL. 2. Se pulsa el botón “Ejecutar Sentencia”.

CU-005	
Nombre:	Exportar el resultado de una consulta a un fichero Excel.
Actores	U
Objetivo	Obtener un documento XML con el resultado de la consulta realizada
Precondiciones	CU-001,CU-002
Poscondiciones	-
Escenario básico	1. 1. Se abre la opción Archivo del menú de herramientas 2. Se pulsa la opción “Exportar a Excel”. 3. Se selecciona Archivo Excel.
Escenario alternativo	

CU-006	
Nombre:	Exportar el resultado de una consulta de Excel a hoja de Excel.
Actores	U
Objetivo	Obtener una hoja de trabajo de Excel rellena con los datos de la consulta realizada.
CU-001,CU-002	CU-001,CU-002
Poscondiciones	-
Escenario básico	1. 1. Se abre la opción Archivo del menú de herramientas 2. Se pulsa la opción “Exportar a Excel”. 3. Se selecciona Archivo Excel.
Escenario alternativo	

CU-007	
Nombre:	Cambiar Idioma Interfaz.
Actores	U
Objetivo	Cambiar el idioma de la interfaz.
Precondiciones	-
Poscondiciones	-
Escenario básico	1. 1. Se Selecciona la opción Edición. 2. Se selecciona la opción Cambiar Idioma. 3. Se selecciona el idioma".
Escenario alternativo	

CU-008	
Nombre:	Ejecutar la eliminación de una tabla (Drop).
Actores	U
Objetivo	Borrar una tabla de la base de datos.
Precondiciones	-
Poscondiciones	-
Escenario básico	1. 1. Se introduce la consulta en lenguaje natural. 2. Se traduce. 3. Se ejecuta pulsando el botón "Ejecutar Sentencia".
Escenario alternativo	1. 2. Se introduce la consulta en lenguaje SQL. 2. Se pulsa el botón "Ejecutar Sentencia".

CU-009	
Nombre:	Ejecutar la limpieza del contenido de una tabla (Truncate).
Actores	U
Objetivo	Limpiar todo el contenido de una tabla.
Precondiciones	-
Poscondiciones	-
Escenario básico	4. 1. Se introduce la consulta en lenguaje natural. 5. Se traduce. 6. Se ejecuta pulsando el botón "Ejecutar Sentencia".
Escenario alternativo	3. 2. Se introduce la consulta en lenguaje SQL. 4. Se pulsa el botón "Ejecutar Sentencia".

5. DEFINICIÓN DE INTERFACES DE USUARIO

El objetivo de este apartado es especificar las interfaces que permitirán la comunicación entre usuario y sistema: diseño de pantallas, diálogos, etc. El fin es conseguir diseñar una interfaz intuitiva, amigable, eficiente y sencilla de utilizar, que satisfaga todos los requisitos definidos, teniendo siempre en cuenta el perfil de usuario al que irá destinado el producto.

Para esto, se especifica qué tipo de información va a necesitar tener en pantalla el usuario para completar su tarea, descomponiendo en pantallas cada uno de los pasos que requiera para ello. Seguidamente se definirá el formato, estilo y contenido de cada una de las interfaces de pantalla definidas, tomando como punto de partida la descripción de cada escenario de los casos de uso.

Además, se pueden seguir los siguientes pasos para conocer el perfil del usuario en cuestión:

- Conocer al usuario y crear un perfil.
- Aplicar los principios de diseño de diálogos.
- Seleccionar el tipo adecuado de ventanas.
- Aplicar los estándares de diseño W3C de los interfaces.
- Definir los menús y subventanas.
- Seleccionar los controles que mejor se acoplen a la interfaz.
- Organizar y presentar las ventanas.
- Crear iconos identificativos.

Estos pasos se desarrollan en las tareas que componen esta actividad y se completarán en el proceso de diseño.

Los sucesivos apartados serán los que definan completamente el diseño de la interfaz de usuario:

- Principios generales de la interfaz.
- Catálogo de perfiles de usuario.
- Controles y elementos de diseño.
- Prototipo.

5.1. ESPECIFICACIÓN DEL INTERFACES GENERALES

La interfaz de usuario se desarrollará con un entorno gráfico WPF con su correspondiente código XAML. Esta interfaz dispondrá de los elementos gráficos (ventanas, listas, botones, barras de herramientas, etc.) necesarios para permitir que el usuario interactúe con el sistema y pueda realizar las tareas para las cuales está diseñado.

DIRECTRICES DE LA INTERFAZ GENERAL.

La W3C establece una serie de patrones de diseño definidos por características mínimas y unas reglas de diseño que limitan la libertad del usuario en cuanto al diseño de interfaces.

Por ello, se seguirán pautas recomendadas por el W3C. A continuación se muestran algunas de los estándares XAML y sus similitudes con otros modelos de especificaciones, CSS2[22], XHTML[23], que aplican el estándar W3C:

XAML	CSS2	Descripción
Italic	Font-style:italic	Texto en cursiva.
ScrollViewer	Scroll	Soporte para el desplazamiento cuando el contenedor es más pequeño que el contenido.

ASI.Tabla 6.- Similitud estándares XAML con W3C.

[22] http://www.w3.org/TR/CSS21/. Último acceso 19-05-2011.

[23] http://www.w3.org/TR/xhtml1/. Último acceso 19-05-2011.55

XAML tag	XHTML tag	XUL tag
Button	Button	Button
Cell	cell	
checkbox	Input type = “checkbox”	checkbox
Frame	Frame, iframe	iframe
Table	Table	grid

ASI.Tabla 7. Similitud tags XAML con tags estandarizados por el W3C.

A continuación se muestra una lista de estándares definidos por el documento guía de mejora de experiencia e Interacción de usuarios propuesta por Microsoft para construir una buena GUI. Se han diferenciado las siguientes categorías y estándares:

CONTROLES

- Descripciones emergentes: Informarán al usuario de problemas no críticos o situaciones especiales.
- CheckBox: Permitirá al usuario decidir entre al menos dos opciones.
- Button: Permitirá al usuario realizar acciones inmediatas.
- GroupBox: Permitirá al usuario ver la relación existente entre los elementos desplegados.
- ListBox: Mostrará los valores en una lista, siempre visible.
- ListView: Permitirá al usuario interactuar con una colección de objetos de datos usando selección simple o múltiple.
- RadioButton: Deberá tener selección exclusiva.
- Slider: Permitirá al usuario seleccionar un rango de valores continuo y regular.
- Tab: Tendrá una etiqueta que la identificará.
- TextBox: Permitirá mostrar, introducir o editar valores tanto alfanuméricos como numéricos.
- TreeView: Permitirá al usuario interactuar con la jerarquía de objetos mostrados, permitiendo selección múltiple.
- Tooltip/Infotip: Mostrará un texto identificador del elemento y atajos de teclado si procede.

COMANDOS

- Menú: Son una lista de comandos u opciones disponibles. Se expandirán al recibir un clic de ratón y lo harán verticalmente hacia abajo. Si es necesario, incluirán submenús en cascada. Los elementos del menú deberán estar capitalizados y permitirán atajos de teclado.
- ToolBar: Podrán dar acceso a opciones del menú directamente. Deberán ser auto explicativas y se agruparán en función de su grupo de comandos básico. Han de ser distinguibles entre sí.
- Ribbon: Organiza comandos en Tabs situados en la parte superior de la ventana. Deberá ser visible, estar etiquetado y no ser jerárquico.

TEXTO

- Deberá identificar la ventana.
- El texto de los Label deberá estar al lado del control al que se asocie.
- Deberá ser preciso, claro, omitir adverbios innecesarios y usar verbos simples.
- Evitar coloquialismos.
- Usar la segunda persona para que el usuario reciba directamente el contexto.
- Usar frases pasivas.
- Ser educado.
- Justificar el texto.

MENSAJES

- Errores y avisos:

    - No reportar mensajes de error sin contenido útil para el usuario. No le digas qué error ha sucedido, dile cómo solucionarlo.
    - Reportar errores que no tengan que ver con el código.
    - Evitar una sobre informar al usuario acerca del error.

- Confirmaciones:

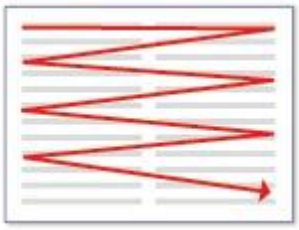
- Obligar al usuario al usuario a pensar acerca de lo que va a confirmar o rechazar; que no decida sin hacerlo previamente.
- Proporcionar toda la información necesaria.

VENTANAS

- Llevar título.
- Usar diseños estándar, sin recargarlas.
- Tener un tamaño adecuado.
- Tener una posición adecuada.
- Permitir maximizar, minimizar y cerrar.

VISUALES

- Jerarquizar la información del modo arriba-abajo e izquierda-derecha.



ASI.Figura 4. Jerarquía lógica de información

- Usar el espacio de la pantalla eficientemente.
- Permitir cambiar el tamaño.
- Usar el foco de controles para que el usuario sepa dónde se encuentra en cada momento dentro de la ventana.
- Usar agrupaciones de elementos cuya lógica relacional sea similar.
- Usar separadores: Son líneas que separan horizontal o verticalmente grupos de controles, proporcionando una vista más clara de los elementos.
- Meter los GroupBox dentro de un marco que los agrupe.
- Usar fondos para enfatizar diferentes tipos de contenido.
- Alinear elementos horizontal y verticalmente y justificarlos.
- Alinear elementos a la izquierda. Los números tendrán una alineación derecha.

5.2. CATALOGO DE PERFILES DE USUARIO

Los perfiles de usuario consisten en un modelo que engloba un conjunto de características, responsabilidades y/o capacidades que puede tener un usuario del sistema. En función a su grado de responsabilidad en cuanto a las funciones que desempeña, y sus conocimientos técnicos, y al nivel de restricción del sistema, se puede inferir que el catálogo de perfiles es el siguiente:

- **Usuario**: Será cualquier individuo que haga uso del sistema.

5.3. COMPORTAMIENTO DINÁMICO DE LA INTERFAZ

No aplicable ya que la pantalla estará constituida por una única ventana de aplicación.

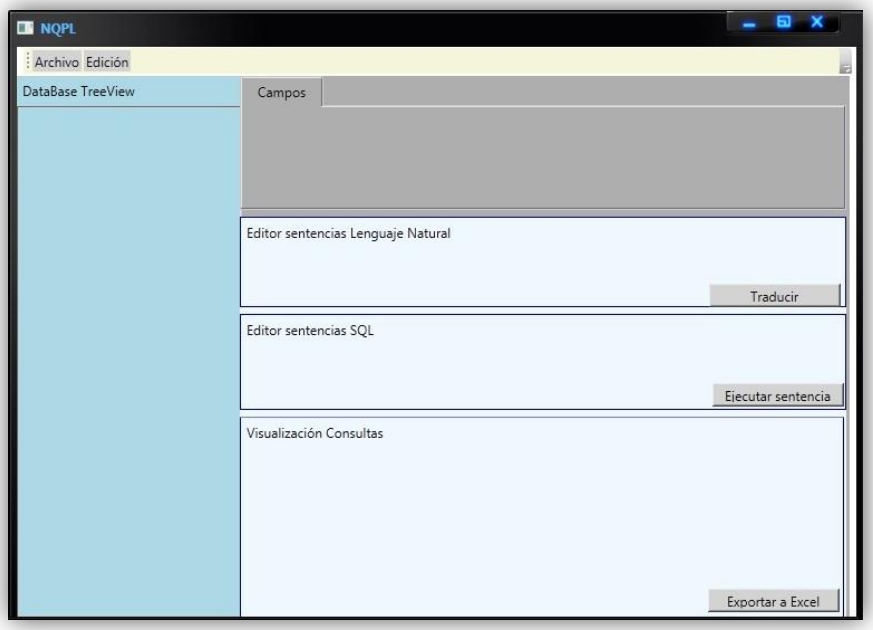
5.4. CATÁLOGO DE CONTROLES Y ELEMENTOS DE DISEÑO

En este apartado se especifica el formato que cada pantalla tendrá individualmente y su posicionamiento, sin tener en cuenta el carácter dinámico que puedan tener. Entre los elementos principales que hay que definir de cara al futuro diseño se distinguen:

- Se podrá modificar el tamaño de la ventana y, nativamente, tendrá una resolución de 1024x768.
- El contenido principal de la ventana deberá situarse en la zona más centrada dentro del marco de la ventana. Los elementos secundarios, como menús, ventanas auxiliares, barras de herramientas, se situarán en la parte izquierda y superior del elemento principal.
- Los únicos elementos de interacción con la ventana de aplicación necesarios serán un ratón y un teclado.
- El archivo exportado de consultas será en formato .xsl o .xslx y se guardará donde el usuario decida.

5.5. PROTOTIPO DE INTERFAZ

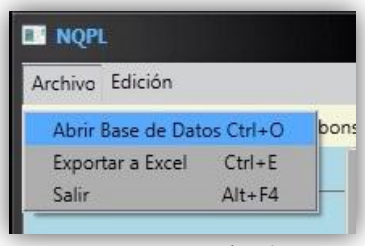
En este apartado se presentará un prototipo muy esquemático de cómo será la interfaz de ventana de la aplicación NQPL.



ASI.Figura 5. Prototipo de NQPL.

Se puede observar que se trata de una interfaz sencilla, en la que determina la posición de los elementos básicos que se van a mostrar.

- **Menú de herramientas**: Incluye la opción de Archivo, mientras que el ítem Edición, meramente representativo, no incluirá en la aplicación. La opción archivo a su vez tendrá las opciones Abrir base de datos, Exportar a Excel y Salir.



ASI.Figura 6. Menú Archivo.

- **DataBase TreeView**: Se mostrará en una jerarquía la base de datos que se ha abierto y el conjunto de tablas que tiene definidas. Dispondrá de un *Scroll* vertical y horizontal por si el contenido no pudiera mostrarse en el espacio reservado para el TreeView.
- **TabControl**: Contendrá un pestaña que mostrará todos los campos pertenecientes a la tabla seleccionada en el *TreeView*. El formato de visualización será como el de un *DataTable*, con las columnas y las tuplas.
- **Espacio del editor Lenguaje Natural**: En este espacio se insertará la sentencia en lenguaje natural que se quiera procesar. Pulsando sobre el botón traducir, se generará en el espacio de sentencias SQL la sentencia traducida para posibles comprobaciones.
- **Espacio del editor de sentencias SQL**: Esta zona está reservada para mostrar la sentencia SQL obtenida a partir de la sentencia original y el módulo traductor. Además, permitirá editar las sentencias traducidas desde el espacio de LN o permitir a usuarios expertos crear sentencias SQL directamente sin pasar por el módulo traductor.
- **Espacio de resultados de consultas**: Mostrará el resultado obtenido al ejecutar la sentencia SQL previamente generada. En el caso de tratarse de una sentencia de actualización, inserción o borrado, mostrará el mensaje resultante, tanto de ejecución correcta como de error, de la sentencia.
- **Mensajes de error y alertas**: En cuanto a los mensajes de alerta y error (como errores al parsear, sentencias incorrectas, etc.), se utilizará toda la zona de la interfaz para, mediante ventanas emergentes, notificar al usuario eventos de este tipo.

6. ANÁLISIS DE CONSISTENCIA Y ESPECIFICACIÓN

El objetivo de esta actividad es la de garantizar la calidad de los modelos generados durante el ASI y certificar que los desarrolladores y los usuarios coinciden en la concepción del sistema final. Para conseguir esto, se genera una matriz de trazabilidad que contendrá los elementos que se comprobarán y verificarán. Si existe asociación entre el elemento columna y el elemento fila, se marca.

6.1. MATRIZ DE TRAZABILIDAD ENTRE REQUISITOS FUNCIONALES Y CASOS DE USO

CU / RS-FUN	001	002	003	004	005	006	007	008	009
RS-FUN-001									
RS-FUN-002	X								
RS-FUN-003			X						
RS-FUN-004			X					X	
RS-FUN-005			X	X				X	X
RS-FUN-006			X	X				X	X
RS-FUN-007					X	X			
RS-FUN-008				X					X
RS-FUN-009				X					
RS-FUN-010							X		
RS-FUN-011									
RS-FUN-012									
RS-FUN-013									
RS-FUN-014						X			
RS-FUN-015									
RS-FUN-016		X							
RS-FUN-017								X	

ASI.Tabla 8. Matriz de trazabilidad entre requisitos funcionales de usuario y casos de uso.

6.2. MATRIZ DE TRAZABILIDAD ENTRE REQUISITOS DE USUARIO Y REQUISITOS DE SOFTWARE

RS / RUC/R	001	002	003	004	005	006	007	008	009	010	011	012
RUC001			X	X	X	X						X
RUC002			X	X	X	X			X			X
RUC003					X	X						X
RUC004			X	X								
RUC005										X		
RUC006					X							
RUC008												
RUC007									X			
RUC009										X		
RUC010												
RUC011				X								
RUC012							X					
RUC013							X					
RUC014												
RUC015			X					X	X			
RUC016												
RUC017									X			
RUC018												
RUC019											X	
RUR01		X										
RUR02			X						X			
RUR03												
RUR04			X									
RUR05												
RUR06												
RUR07	X											
RUR08												
RUR09												
RUR10												
RUR11												
RUR12												
RUR13												
RUR14												
RUR15												
RUR16												
RUR17												
RUR18												
RUR19												

ASI.Tabla 9. Matriz de trazabilidad entre requisitos de usuario y software I.

RS / RUC/R	013	014	015	016
RUC001				X
RUC002				X
RUC003				X
RUC004				
RUC005				
RUC006				
RUC007				
RUC009				
RUC010				
RUC011				
RUC012				
RUC013				
RUC014				
RUC015				
RUC016		X		
RUC017			X	
RUC018		X		
RUC019	X			
RUR01				
RUR02				
RUR03				
RUR04				
RUR05				
RUR06				
RUR07				
RUR08				
RUR09				
RUR10				
RUR11				
RUR12				
RUR13				
RUR14				
RUR15				X
RUR16				X
RUR17				X
RUR18				X
RUR19				

ASI.Tabla 10.Matriz trazabilidad entre requisitos de usuario y software II.

7. DEFINICIONES Y ACRÓNIMOS

7.1. DEFINICIONES

Actor Entidad externa al sistema que demanda una funcionalidad de este.

Button Control XAML que está dotado de funcionalidad.

Caso de uso Secuencia de interacciones entre un actor y un sistema de información.

CheckBox Elemento que permite hacer elecciones múltiples sobre un conjunto de opciones.

Cell Cada uno de los elementos de una tabla.

Clic Acción que incluye algún botón del ratón.

Foco Concepto que define el estado de un objeto cuando recibe una entrada de teclado o ratón.

Frame Marco o cuadro.

Infotip Pequeña ventana de información que aparece sobre un objeto que no está etiquetado.

Label Etiqueta de texto de un control que proporciona soporte a los atajos de teclado.

ListBox Control desplegable que muestra una lista de objetos.

ListView Control que muestra una colección de objetos.

Microsoft Excel Aplicación de Microsoft incluida en la suite Office que permite manejar hojas de cálculo.

RadioButton Control que permite seleccionar una única opción dentro de un grupo de opciones.

Slider Control que permite al usuario seleccionar un valor en un rango deslizando un tirador a lo largo de un carril.

Tab Control que permite definir múltiples páginas dentro del mismo área de ventana.

TabControl Control que gestiona los tabs.

TextBox Control que se usa para mostrar o editar texto.

TreeView Control que muestra una colección jerárquica de objetos etiquetados.

ToolBar Control que sirve como contenedor a otros controles.

Tooltip Control que representa una ventana emergente de pequeño tamaño y que muestra información acerca del control al que está ligado.

GroupBox Control que muestra un marco alrededor de un grupo de controles.

Scroll Desplazamiento.

ScrollViewer Área desplazable que puede contener otros objetos.

Table Tabla. Elemento que contiene información distribuida y organizada en columnas y filas.

Tag Etiqueta de marcado.

.xsl Extensión para los archivos de Microsoft Excel.

7.2. ACRÓNIMOS

CSS2 Cascade Style Sheet (Hoja de Estilo en Cascada).

XHTML eXtensible HyperText Markup Language (Lenguage Extensible de Marcado de Hipertexto).

XUL XML based User interface Language (Lenguaje XML basado en Interfaces de Usuario).

XSL eXtensible Stylesheet Language (Lenguaje de Hijas de Estilo Extensible).

8. REFERENCIAS

[1] *Carlos Cócera Pérez. "Diseño y Codificación de un Complemento Multilenguaje Integrado en Paquetes de Productividad para Facilitar la Transformación del Lenguaje Natural en Sentencias de Interacción con Bases de Datos". Estudio de Viabilidad del Sistema. Proyecto de Fin de Carrera. Último acceso 29-06-2011.*

[2] *UML Resource Page. (http://www.uml.org/). Último acceso 05-06-2011.*

[3] *Consejo Superior de Administración Electrónica.- MÉTRICA. VERSIÓN 3 Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información. Último acceso 03-06-2011.*

[4] *Microsoft MSDN (http://msdn.microsoft.com/es-es/ms348103). Último acceso 28-07-2011.*

[5] *Donald Bell, "UML basics: An introduction to the Unified Modeling Language", 2003 Rational Software. Último acceso 11-06-2011. (http://doc.orizondo.org/refs/uml/UMLBasics.zip)*

[6] *R.A.E. (http://www.rae.es/rae.html). Último acceso 28-06-2011.*

Wikipedia. (http://es.wikipedia.org/wiki/Wikipedia:Portada). Último acceso 28-06-2011

APÉNDICE C

PLAN DE GESTIÓN DEL PROYECTO

RESUMEN

En este apéndice se pretende definir el Plan de Gestión del Proyecto. En él se va a describir la técnica empleada en la administración del proyecto de implementación de un traductor de sentencias en LN a SQL.

En el documento se recogerá una visión de cuál va a ser la organización del proyecto en cuanto a límites organizativos y roles, así como al proceso técnico que definirá las entradas y salidas del proyecto y herramientas empleadas en el proceso. Se establecerá una planificación temporal y un presupuesto estimado para el proyecto. Los objetivos serán por tanto controlar que le proyecto vaya según lo esperado (plan) y si no es así (seguimiento) tomar las decisiones adecuadas (control).

HOJA DE ESTADO DEL DOCUMENTO

NQPL		
GESTIÓN DEL PROYECTO. HOJA DE ESTADO DEL DOCUMENTO		
Versión	**Fecha**	**Motivación**
0.01	21-06-2011	Primera versión

PGP.Tabla 1. Hoja de estado del documento.

REGISTRO DE CAMBIOS

NQPL		
GESTIÓN DEL PROYECTO. HISTORIAL DE CAMBIOS		
Versión	**Fecha**	**Motivación**
0.01	21-06-2011	Primera versión

PGP.Tabla 2. Historial de cambios.

NQPL			
GESTIÓN DEL PROYECTO. VALIDACIÓN DEL DOCUMENTO			
Versión	**Fecha**	**Cargo**	**Nombre y Apellidos**
0.01	21-06-2011	Jefe de Proyecto	Carlos Cócera Pérez

PGP.Tabla 3. Validación del documento.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. GESTIÓN DEL PROYECTO

1.1. INTRODUCCIÓN

El presente documento pretende describir el Plan de Gestión del Proyecto asociado al desarrollo de una aplicación que permita insertar sentencias en lenguaje natural, las traduzca al lenguaje SQL y genere el resultado de la consulta.

Lo que se va a recoger en este documento es la administración del plan de desarrollo de la aplicación así como del propio documento de gestión de proyecto.

1.2. PROPÓSITO DEL DOCUMENTO

Este documento trata de definir los objetivos, qué entregables se van a generar y el ciclo de vida de la aplicación NQPL. A su vez, definirá un catálogo de actividades que marcarán el curso del desarrollo, los recursos necesarios, una planificación temporal y el presupuesto destinado a la consecución.

1.3. ÁMBITO DE SOFTWARE

Tal y como se ha definido en capítulos anteriores, el objetivo del software que se va a desarrollar es la traducción a lenguaje SQL de una sentencia en lenguaje natural. El motivo de este proyecto es el resultado de la necesidad tecnológica de acercar al usuario y la máquina, en beneficio de la misma. La aplicación se implementará a través de la plataforma de desarrollo .NET, por popularidad del sistema operativo que la sostiene; el código se generará en lenguaje C# y la interfaz en formato WPF.

1.4. ESTRUCTURA DEL DOCUMENTO

El presente documento guarda la siguiente estructura en cuanto a contenido:

- **Bloque 1**: Incluye la introducción, definición y objetivos del Plan de Gestión del Proyecto.
- **Bloque 2**: Consta de la gestión del proyecto en cuando a roles y límites de la estructura organizacional.
- **Bloque 3**: Consiste en el desarrollo de un proceso técnico en el que se detallarán tanto las entradas y salidas del proyecto, como las herramientas empleadas para obtenerlas.
- **Bloque 4:** Mostrará la planificación temporal del proyecto y una estimación del presupuesto necesario para este proyecto.

1.5. ORGANIZACIÓN DEL PROYECTO

Este apartado define la estructura de gestión del proyecto y delimita las responsabilidades de cada uno de los miembros participantes.

ROLES ORGANIZATIVOS Y RESPONSABILIDADES

Ya se han definido en el Estudio de Viabilidad del Sistema, "*3.2. IDENTIFICACIÓN DE LOS USUARIOS PARTICIPANTES EN EL ESTUDIO DE LA SITUACIÓN ACTUAL".*

LÍMITES ORGANIZATIVOS E INTERFACES

El presente apartado ya ha sido definido en el Estudio de Viabilidad del Sistema, "*2. DEFINICIÓN DEL SISTEMA*".

1.6. PROCESO TÉCNICO

En este bloque se establecerán los detalles técnicos del proyecto, los cuales incluyen las entradas y salidas del proyecto, así como las herramientas empleadas para la consecución del mismo.

ENTRADAS DEL PROYECTO

La principal entrada del proyecto ha sido la necesidad tecnológica de acercamiento lingüístico entre el hombre y la máquina como componente integrado en suites de productividad. Ciertamente se pueden considerar entradas del proyecto las referencias obtenidas de metodologías y técnicas aplicadas en proyectos ya existentes para la obtención del proceso de traducción natural a lenguaje formal.

SALIDAS DEL PROYECTO

La salida principal del proyecto es la aplicación traductora de lenguaje natural a sentencias SQL la cual consta de los componentes detallados en el Apéndice A "*7.2. DESCRIPCIÓN DE LOS COMPONENTES DE LA SOLUCIÓN*".

TECNOLOGÍA Y HERRAMIENTAS DE DESARROLLO

Durante el proceso de desarrollo de este proyecto se han aplicado las tecnologías y herramientas más vanguardistas y populares del mercado con el fin de obtener un producto final sólidamente cimentado y lejano de ser arcaico. Documentación más destallada se puede encontrar en el Capítulo III, apartados "*4.3. TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO DE* NQPL" y "*4.4. HERRAMIENTAS EMPLEADAS EN EL* DESARROLLO".

FUNCIONES DE APOYO AL PROYECTO

Debido al tamaño del proyecto y los pocos recursos para su desarrollo, no se ha elaborado el documento de gestión de configuración, validación y aseguramiento de calidad.

2. PLANIFICACIÓN TEMPORAL

En este documento se pretende definir la planificación y costes del proyecto NQPL. En la planificación se descompondrán las tareas del proyecto y su evolución temporal a lo largo de la ejecución del mismo.

2.1. CALENDARIO

Resulta crucial e indispensable hablar de una planificación precisa y adecuada si se quiere conseguir alcanzar un resultado idóneo.

Obviamente, existen posibilidades de que la planificación sufra modificaciones a lo largo del ciclo de desarrollo de un proyecto, pero hay que estar prevenido para adaptarse a dichas alteraciones.

Particularmente en este proyecto, la planificación inicial fue buena, pero por circunstancias ajenas al equipo de desarrollo, de carácter no presupuestario, se congeló el análisis y diseño y se detuvo el desarrollo. Una vez reiniciado el proyecto, la mayor parte del tiempo hubo que dedicarlo al análisis de las tecnologías y metodologías existentes para analizar lenguaje natural y construir sentencias SQL. Sin embargo, otras tareas no requirieron de tanto tiempo de la planificación, con lo que el proyecto se ha podido finalizar justo a tiempo. Entre las tareas que se llevaron a cabo en el desarrollo del proyecto se denominan tareas y se definen las siguientes:

- **La primera tarea** consiste en el estudio y documentación aquellas líneas de investigación que están abiertas dentro del ámbito de la ingeniería de la información y que estaban siendo objeto de trabajo en la actualidad. Una vez seleccionado cuál iba a ser objetivo el producto y por qué línea se iba a encarrilar el proyecto, se pasa la fase familiarización con esta vía de investigación, en cuanto a tecnologías y herramientas usadas y proyectos referentes similares desarrollados.
- **La segunda tarea** Consiste en un análisis de cómo se va construir el parser del lenguaje natural a lenguaje SQL. Se analizan los posibles sistemas de análisis

gramatical y se decide cuál va a ser, en función de las capacidades del equipo de desarrollo y presupuestarias, el sistema que mejor se ajuste a las necesidades del proyecto.

- **La tercera** y última tarea, ocupa más del 75% del tiempo y consiste en la implementación de los componentes de traducción deseados y previamente analizados, de la implementación del código necesario para dotar a la aplicación de la funcionalidad requerida por el cliente, de la generación de la interfaz de comunicación entre usuario y computadora y de la documentación final del producto.

En el siguiente apartado se muestra la planificación de las tareas de las que consta el proyecto NQPL donde se recogerá una relación de las tareas, sus fechas de inicio y finalización estimadas.

2.2. PLANIFICACIÓN

		Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**PROYECTO NQPL**	**130 días**	**vie 07/01/11**	**jue 07/07/11**	
2	✓	**ANÁLISIS Y ESPECIFICACIÓN DE**	**9 días**	**vie 07/01/11**	**mié 19/01/11**	
3	✓	Estudio previo y entrevistas co	2 días	vie 07/01/11	lun 10/01/11	
4	✓	Identificación de requisitos de u	2 días	mar 11/01/11	mié 12/01/11	3
5	✓	Identificación de requisitos SW	1 día	jue 13/01/11	jue 13/01/11	4
6	✓	Identificación de requisitos de l	1 día	vie 14/01/11	vie 14/01/11	5
7	✓	Preparación de la documentaci	1 día	lun 17/01/11	lun 17/01/11	6
8	✓	Documentación general	1 día	mar 18/01/11	mar 18/01/11	7
9	✓	Validación de la documentación	1 día	mié 19/01/11	mié 19/01/11	8
10		**ADAPTACIÓN AL ENTORNO Y AI**	**2 días**	**mié 19/01/11**	**vie 21/01/11**	**2**
11		Visual Studio 2010	1 día	mié 19/01/11	jue 20/01/11	
12	✓	SQL	1 día	jue 20/01/11	jue 20/01/11	
13	✓	Firebird	1 día	vie 21/01/11	vie 21/01/11	12
14	✓	XAML	1 día	jue 20/01/11	jue 20/01/11	
15	✓	WPF	1 día	jue 20/01/11	jue 20/01/11	
16	✓	**MODELADO CONCEPTUAL**	**13 días**	**lun 24/01/11**	**mié 09/02/11**	**10**
17	✓	Construcción del modelo de Ob	4 días	lun 24/01/11	jue 27/01/11	
18	✓	Especificación de los métodos	7 días	vie 28/01/11	lun 07/02/11	17
19	✓	Revisión de la tarea	1 día	mar 08/02/11	mar 08/02/11	18
20	✓	Validación del documento	1 día	mié 09/02/11	mié 09/02/11	19

PGP.Figura 1. Planificación del proyecto.

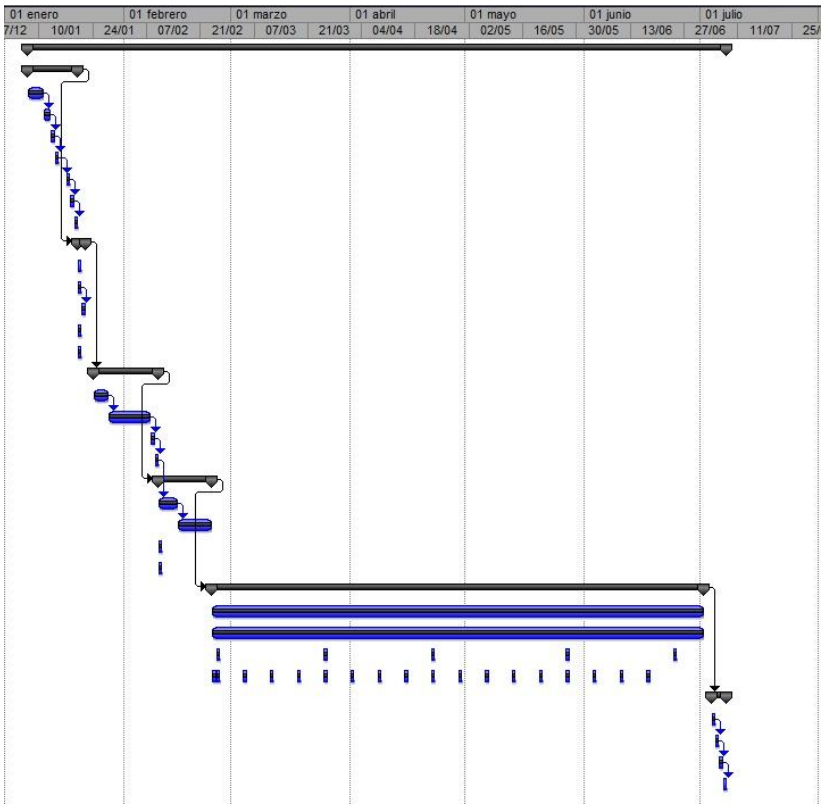
Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
26	**CODIFICACIÓN**	**92 días**	**jue 24/02/11**	**vie 01/07/11**	**21**
27	Implementación de la aplicación	92 días	jue 24/02/11	vie 01/07/11	
28	Documentación de la fase de c	92 días	jue 24/02/11	vie 01/07/11	
29	**Pruebas Funcionalidad Prot**	**86 días**	**vie 25/02/11**	**vie 24/06/11**	
30	Pruebas Funcionalidad Prot	1 día	vie 25/02/11	vie 25/02/11	
31	Pruebas Funcionalidad Prot	1 día	vie 25/03/11	vie 25/03/11	
32	Pruebas Funcionalidad Prot	1 día	vie 22/04/11	vie 22/04/11	
33	Pruebas Funcionalidad Prot	1 día	vie 27/05/11	vie 27/05/11	
34	Pruebas Funcionalidad Prot	1 día	vie 24/06/11	vie 24/06/11	
35	**Revisión de Código**	**82 días**	**jue 24/02/11**	**vie 17/06/11**	
36	Revisión de Código 1	1 día	jue 24/02/11	jue 24/02/11	
37	Revisión de Código 2	1 día	vie 25/02/11	vie 25/02/11	
38	Revisión de Código 3	1 día	vie 04/03/11	vie 04/03/11	
39	Revisión de Código 4	1 día	vie 11/03/11	vie 11/03/11	
40	Revisión de Código 5	1 día	vie 18/03/11	vie 18/03/11	
41	Revisión de Código 6	1 día	vie 25/03/11	vie 25/03/11	
42	Revisión de Código 7	1 día	vie 01/04/11	vie 01/04/11	
43	Revisión de Código 8	1 día	vie 08/04/11	vie 08/04/11	
44	Revisión de Código 9	1 día	vie 15/04/11	vie 15/04/11	
45	Revisión de Código 10	1 día	vie 22/04/11	vie 22/04/11	
46	Revisión de Código 11	1 día	vie 29/04/11	vie 29/04/11	
47	Revisión de Código 12	1 día	vie 06/05/11	vie 06/05/11	
48	Revisión de Código 13	1 día	vie 13/05/11	vie 13/05/11	
49	Revisión de Código 14	1 día	vie 20/05/11	vie 20/05/11	
50	Revisión de Código 15	1 día	vie 27/05/11	vie 27/05/11	
51	Revisión de Código 16	1 día	vie 03/06/11	vie 03/06/11	
52	Revisión de Código 17	1 día	vie 10/06/11	vie 10/06/11	
53	Revisión de Código 18	1 día	vie 17/06/11	vie 17/06/11	
54	**PRUEBAS**	**4 días**	**lun 04/07/11**	**jue 07/07/11**	**26**
55	Pruebas Individuales	1 día	lun 04/07/11	lun 04/07/11	
56	Pruebas de integración	1 día	mar 05/07/11	mar 05/07/11	55
57	Prueba del sistema	1 día	mié 06/07/11	mié 06/07/11	56
58	Pruebas de aceptación del clie	1 día	jue 07/07/11	jue 07/07/11	57

PGP.Figura 2. Planificación del proyecto.

2.3. DIAGRAMA DE GANTT

Se muestra en el presente apartado el diagrama de Gantt asociado a la planificación donde claramente se puede observar la evolución en el tiempo de cada tarea:



PGP.Figura 3. Diagrama de Gantt del proyecto.

2.4. PRESUPUESTO

Una vez obtenida la planificación del proyecto, se procede al cálculo del coste del proyecto en su totalidad.

Para ello, hay que calcular cuántas horas se han trabajado desde el comienzo del proyecto hasta su finalización y baremar el coste de cada una de esas horas. Para este proyecto, el total de horas dedicadas asciende a un total de 130 días de trabajo, distribuidos a lo largo de 6 meses, lo que implica un total de 1040 horas efectivas trabajadas.

Si se establece que el salario medio de un Ingeniero de Software es de unos 18€/h y sin tener en cuenta el salario medio de un Jefe de Proyecto, un Analista y un Administrador de

Bases de Datos, ya que es una persona la que desempeña dichas funciones, el total asciende a 18.720,00 euros.

A esta cantidad habría que sumarle el coste de adquisición del software y hardware necesarios para la implementación de la aplicación.

PRODUCTO	CANTIDAD	TOTAL (€)
Salario ingeniero	1	18.720,00
Windows 7	1	299
Visual Studio 2010	1	12.769
Microsoft Office 2010	1	379,01
Ordenador	1	1.372,40
IVA (16%)		5.745,67
		41.656,12

PGP.Tabla 4. Costes del proyecto.

El total ascendería a 41.565,12€ sobre presupuesto total disponible de 50.000€, es decir, hay una gestión de recursos de costes buena que ciertamente se podría optimizar empleando herramientas libres, como Ubuntu, Mono y LibreOffice.

PRODUCTO	CANTIDAD	TOTAL (€)
Salario ingeniero	1	18.720,00
Ubuntu	1	0
Mono	1	0
LibreOffice	1	0
Ordenador	1	1.372,40
IVA (16%)		3.249,92
		23.561,90

PGP.Tabla 5. Costes proyecto con software gratuito.

La reducción de inversión es sustancial pero a coste popularidad y de estabilidad de arquitectura, lo que a la larga, pasa factura. Pero sí es cierto que la alternativa es suculenta ya

que con el ahorro se podrían contratar más desarrolladores y reducir la planificación hasta 10 veces.

3. DEFINICIONES Y ACRÓNIMOS

3.1. DEFINICIONES

LibreOffice Suite ofimática libre, gratuita y multiplataforma.

Modelo de Constes Constructivos Modelo de estimación de costes y esfuerzo de proyectos software.

3.2. ACRÓNIMOS

IVA Impuesto sobre el Valor Añadido.

4. REFERENCIAS

[1] *Carlos Cócera Pérez. "Diseño y Codificación de un Complemento Multilenguaje Integrado en Paquetes de Productividad para Facilitar la Transformación del Lenguaje Natural en Sentencias de Interacción con Bases de Datos". Estudio de Viabilidad del Sistema. Proyecto de Fin de Carrera. Último acceso 28-06-2011.*

[2] *Consejo Superior de Administración Electrónica.*- MÉTRICA. VERSIÓN 3 Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información. *Último acceso 03-06-2011.*

[3] *Plan de Gestión del Proyecto. Último acceso 08-06-2011. (http://administracionelectronica.gob.es/recursos/pae_000001038.pdf).*

[4] *R.A.E. (http://www.rae.es/rae.html). Último acceso 28-06-2011.*

[5] *Wikipedia. (http://es.wikipedia.org/wiki/Wikipedia:Portada). Último acceso 28-06-2011.*

APÉNDICE D

MANUAL DE USUARIO

RESUMEN

En el presente documento se pretende describir las situaciones que experimenta un usuario cuando pretende usar la aplicación NQPL para realizar operaciones sobre una base de datos.

HOJA DE ESTADO DEL DOCUMENTO

NQPL		
MANUAL DE USUARIO. HOJA DE ESTADO DEL DOCUMENTO		
Versión	**Fecha**	**Motivación**
0.01	03-07-2011	Primera versión

REGISTRO DE CAMBIOS

NQPL		
MANUAL DE USUARIO. HISTORIAL DE CAMBIOS		
Versión	**Fecha**	**Motivación**
0.01	03-07-2011	Primera versión

NQPL			
MANUAL DE USUARIO. VALIDACIÓN DEL DOCUMENTO			
Versión	**Fecha**	**Cargo**	**Nombre y Apellidos**
0.01	03-07-2011	Jefe de Proyecto	Carlos Cócera Pérez

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. DESCRIPCIÓN GENERAL

El proyecto de implementación de un complemento multilenguaje que permita efectuar operaciones sobre una base de datos empleando el lenguaje natural establece que cualquier usuario, independientemente del nivel de conocimientos que posea, pueda interactuar con una base de datos SQL. Para esto el usuario necesitará:

- ✓ Equipo con características mínimas descritas en requisitos de software.
- ✓ Tener instalado Visual Studio 2008 o superior.
- ✓ Tener instalado Microfost Office 2007 o 2010.
- ✓ Tener instalado Firebird .NETProvider-2.6.5 o superior, disponible en http://www.firebirdsql.org/en/net-provider/
- ✓ Tener instalado Firebird-2.5.0.26074_1_Win32.exe client. Se puede obtener en la dirección http://www.firebirdsql.org/en/firebird-2-5/.
- ✓ Disponer de bases de datos Firebird definidas .FBD.
- ✓ Tener instalado un SGBD de Firebird, por ejemplo, Flamerobin, flamerobin-0.9.2-1-setup.exe disponible en http://sourceforge.net/projects/flamerobin/files/flamerobin/.
- ✓ .NET 3.x o superior. Esto implica tener instalados y actualizados Windows XP, Windows Vista o Windows 7 todos ellos con arquitectura 32bits.

2. PROCESO PREPARATIVO DE LA INSTALACIÓN DE LA APLICACIÓN

2.1 INSTALACIÓN DE LA TECNOLOGÍA

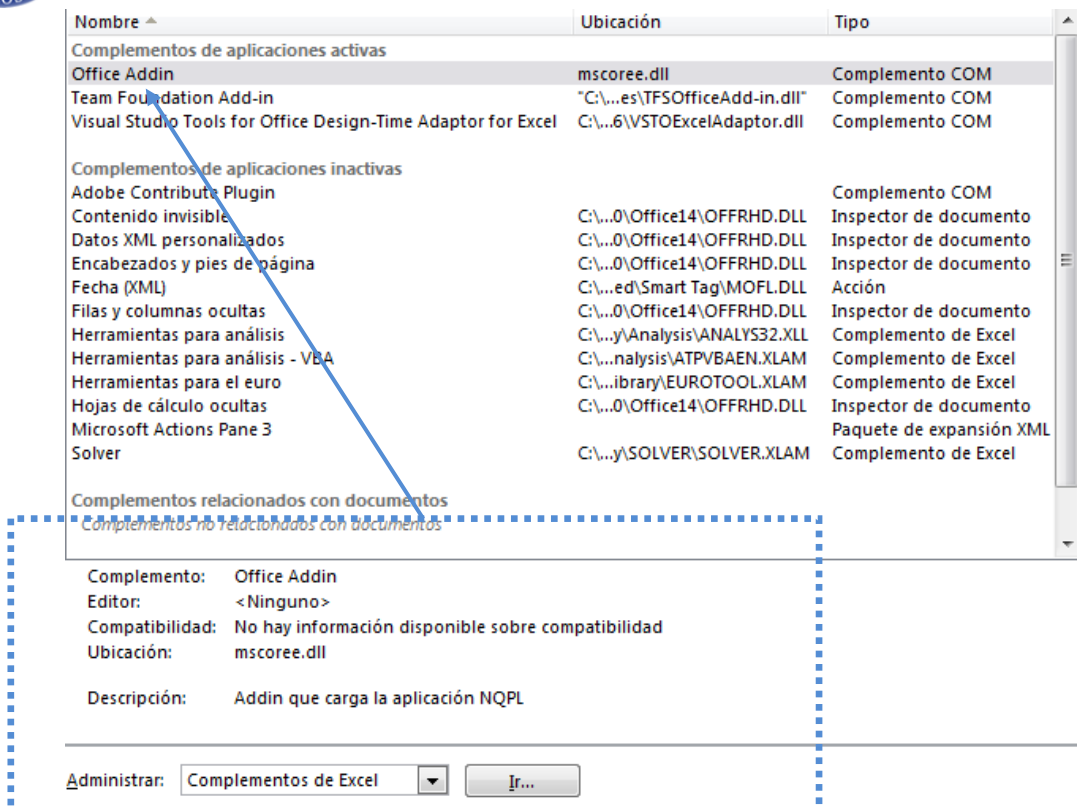
El primer paso es disponer de la tecnología indicada en el apartado 1. Una vez instaladas las tecnologías procederemos a efectuar los siguientes pasos

2.2 EJECUCIÓN DESDE MICROSOFT OFFICE TRAS REGISTRO DEL ADD-IN

En este apartado se va a comentar cómo se ejecutaría la aplicación NQPL desde el complemento COM integrado en Microsoft Office. Puesto que se trata de una aplicación orientada al manejo de base de datos y en ella se han realizado exportaciones de resultados a Excel, la ejecución se explicará desde Microsoft Excel.

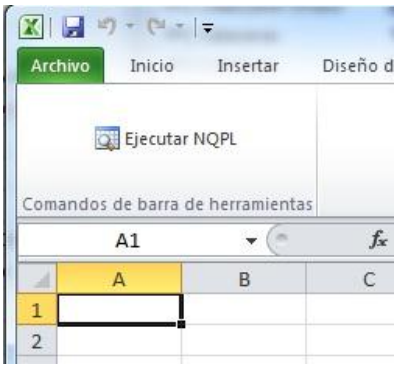
Como se ha comentado anteriormente, el Add-in se ha generado para registrarlo en todos los componentes Office, no es exclusivo de Excel.

El primer paso sería compilar/instalar la solución, tras lo cual se registra el componente COM y se crea el Add-in en Office:



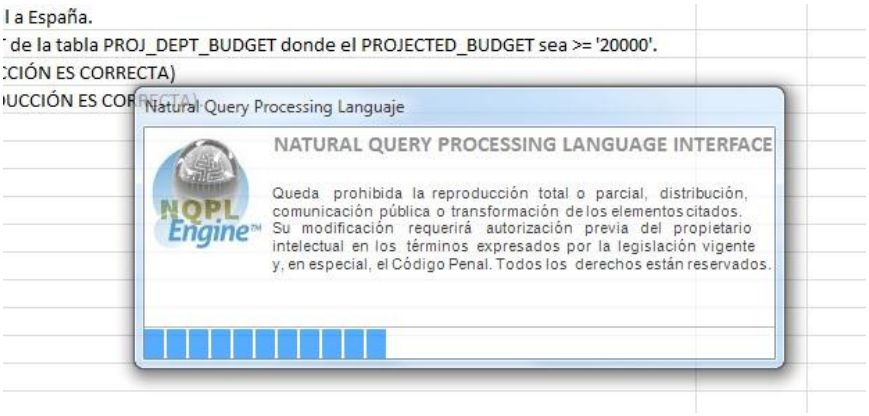
MU. Figura 1. Lista de componentes instalados en Excel 2010.

Una vez cargado el Add-in, podremos cerrar la aplicación Excel y al abrirla, el Add-in se volverá a cargar. Este Add-in se ha configurado en código para que instale en la barra de comandos de las aplicaciones Office un botón llamado “Ejecutar NQPL” que tiene asociado un icono y que lanzará la ejecución de NQPL.



MU. Figura 2. Add-in NQPL

Al accionar el botón del Add-in se cargará la aplicación. Durante el proceso de carga se ha decidido cargar una SplashScreen para evitar que el usuario pueda pensar que la aplicación no está cargando, aunque en este caso, la carga de NQPL es tan rápida que es difícil que puedan pensar eso.



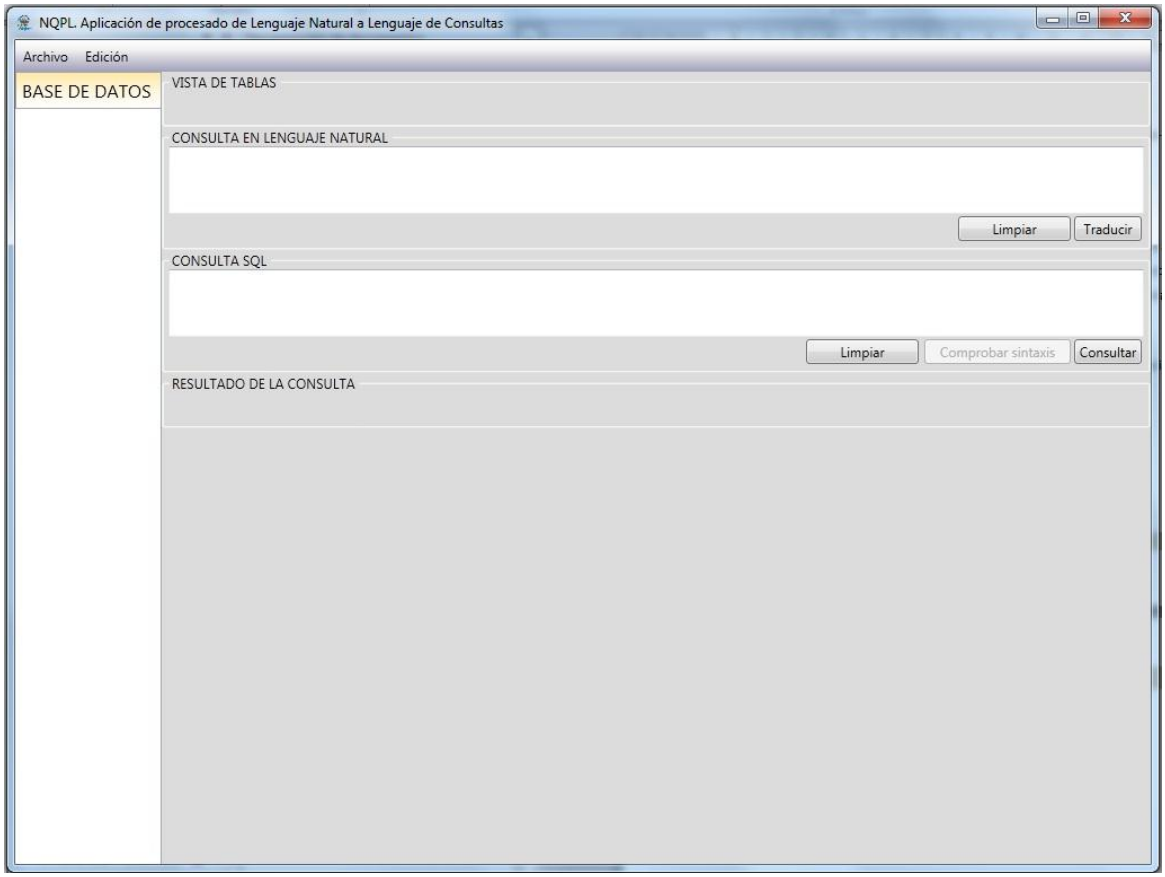
MU. Figura 3. Carga de la SPlashSCreen.

Tras la carga de la aplicación, la SplashScreen desaparecerá y se cargará la ventana principal de NQPL

2.3. LA INTERFAZ NQPL

La interfaz NQPL es bastante sencilla e intuitiva para facilitar al usuario su manejo. Esta ventana consta de las mismas áreas que se definieron en el prototipo inicial pero con algunas mejoras, tanto visuales como funcionales que se han ido recopilando a lo largo de las fases de desarrollo del proyecto.

En este apartado se va a tratar de mostrar las principales funciones de la interfaz así como algunos ejemplos de ejecuciones.

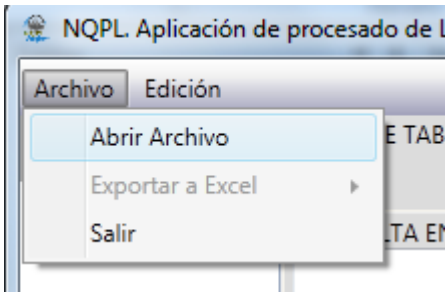


MU. Figura 4. Interfaz NQPL.

Una vez cargada la aplicación estaremos en disposición de efectuar diferentes acciones.

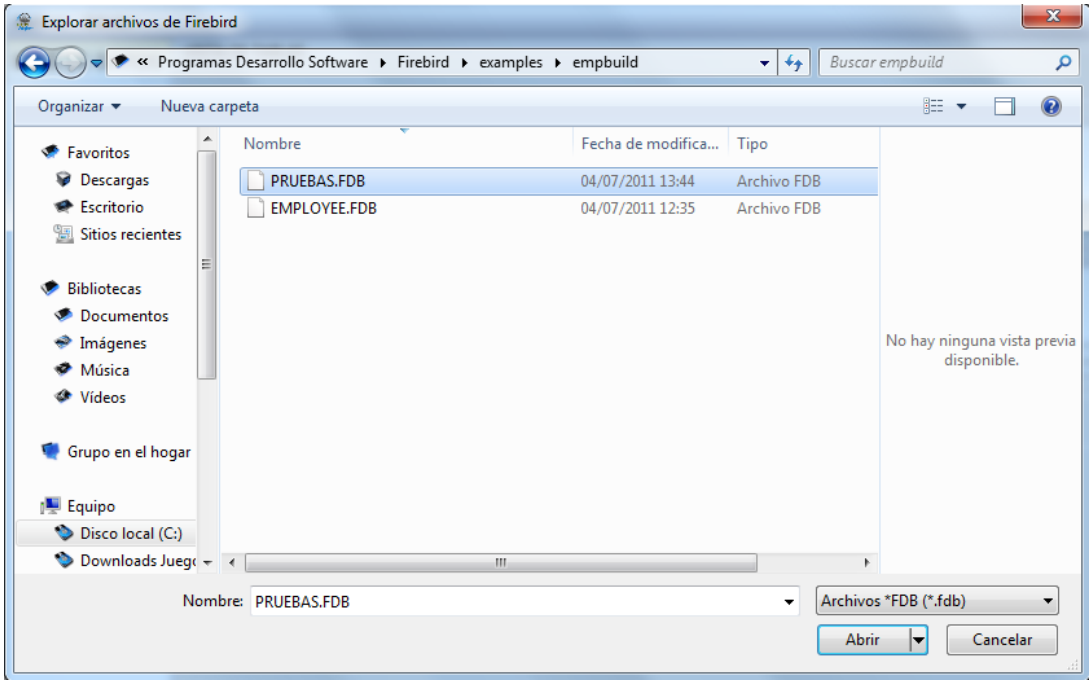
CARGA DE UNA BASE DE DATOS

NQPL permite cargar una base de datos, concretamente bases de datos de Firebird .FBD, sobre la cuales se podrán realizar operaciones. Para ello habrá que ir a la opción del menú Archivo/ Abrir Archivo.



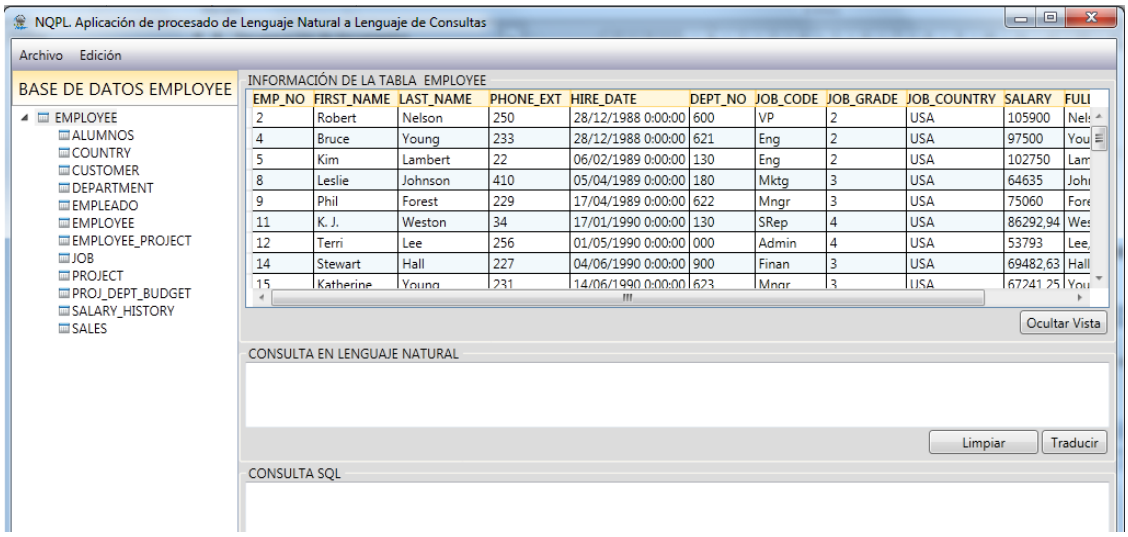
MU. Figura 5. Abrir base de datos I.

Una vez seleccionada la opción se abrirá el explorador de Windows y permitirá al usuario buscar una base de datos para abrir.



MU. Figura 7. Abrir base de datos II.

Obsérvese que se ha configurado para que los archivos que se puedan abrir sean *.FBD. Una vez seleccionada la base de datos que se quiere abrir, se pulsará el botón Abrir y se cargará en la pantalla principal la base de datos con todas las tablas que contiene en forma de árbol jerárquico.

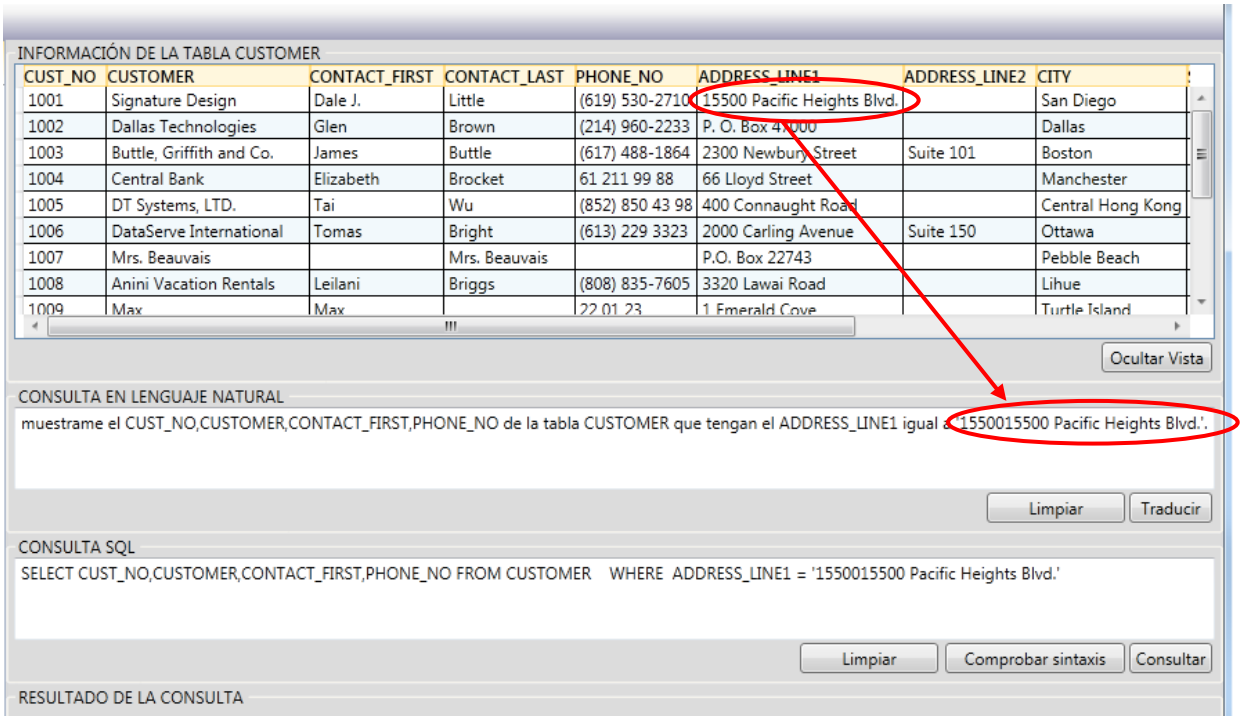


MU. Figura 6. Abrir base de datos III.

Se puede observar que en la base de datos Employee existen varias tablas Alumnos, Country, Customer, etc que pueden seleccionarse para poder ver el contenido de sus campos en la zona del visor de tablas. El siguiente paso sería seleccionar una tabla para realizar la consulta. En este punto habría dos opciones:

Realizar la consulta en lenguaje natural y traducirla a SQL

Para ejecutar una consulta en lenguaje natural, hay que escribir en la zona reservada para ello la consulta. Dicha consulta una vez construida, se podrá traducir a SQL pulsando el botón "Traducir" de la interfaz. La consulta equivalente en SQL se mostrará en el espacio inmediatamente inferior al utilizado. Hay que decir que durante la construcción de la consulta en lenguaje natural, se ha facilitado al usuario la tarea de escribir nombres de tablas, columnas o campos largos y complicados, dándole posibilidad de copiar dichos valores haciendo clic derecho sobre lo que desee copiar.



MU. Figura 8. Traducción de sentencia en Lenguaje Natural.

Realizar la consulta en lenguaje SQL y ejecutarla

Así un clic derecho sobre el campo con valor ‘15500 Pacific Heights Bvd’. agregará al editor de sentencias dicho valoren lenguaje natural para ahorrar tiempo al usuario. Hay que indicar que siempre habrá que cerrar las sentencias con el finalizador ”.” que indicará al parser que la sentencia ha terminado en ese carácter, que no siga interpretando caracteres. Una vez construida la sentencia, se pulsará sobre el botón Traducir y se traducirá el resultado a SQL a la zona inferior.

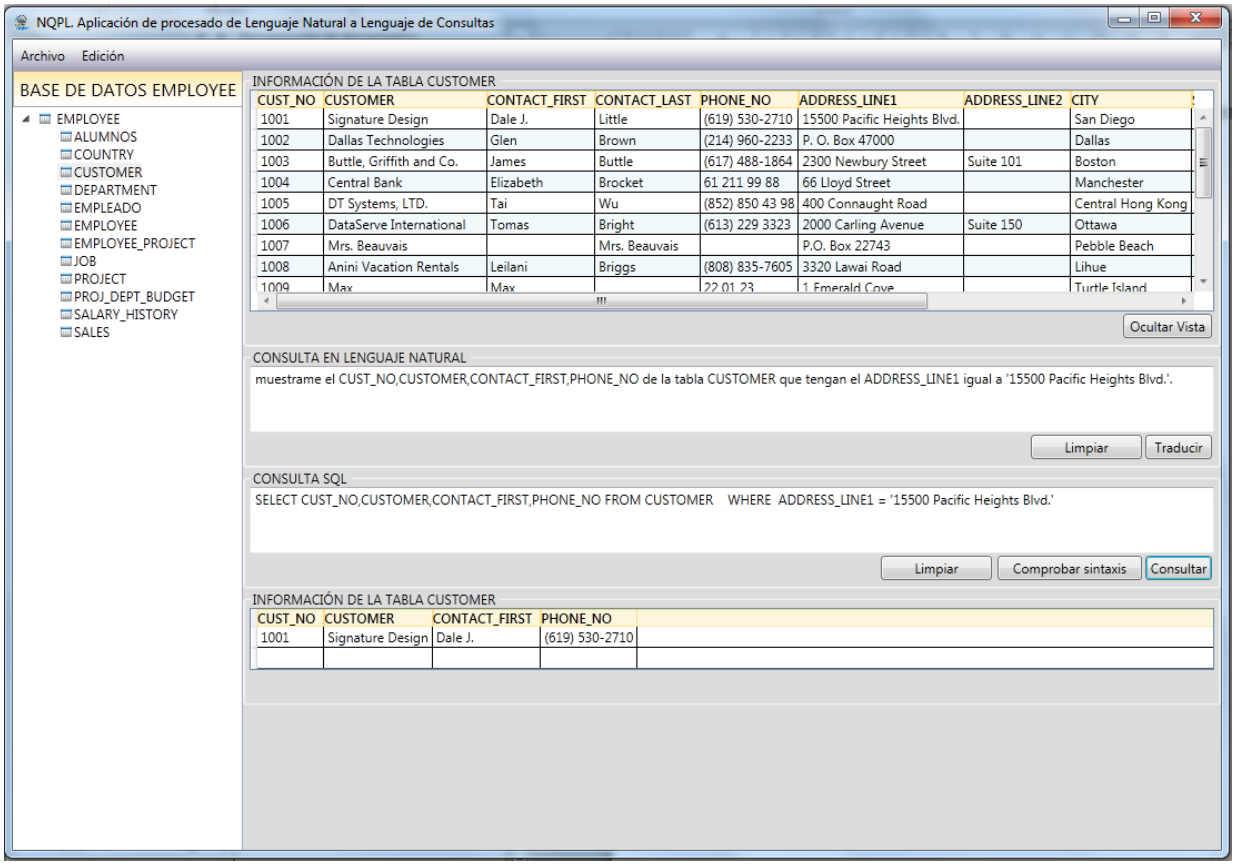
Por si el lector no se ha percatado, uno de los botones de la zona del editor SQL, concretamente el botón “Comprobar sintaxis”, previamente estaba deshabilitado y tras traducir se ha habilitado; la lógica está clara.

En este punto convergen esta opción y la siguiente, por lo que continuaremos en el apartado que sigue a continuación.

Realizar la consulta en lenguaje SQL y ejecutarla

Una vez que tenemos la sentencia en SQL podemos pedirle a la aplicación que compruebe el resultado antes de lanzar a la base de datos la consulta. Esto tiene su lógica ya que se consigue ahorrar lanzar operaciones erróneas a la base de datos. Si la sentencia es correcta no aparecerá ningún mensaje de error. Tras esto, el usuario podrá ejecutar la consulta y, si hay resultados, se mostrarán en la zona reservada para ello.

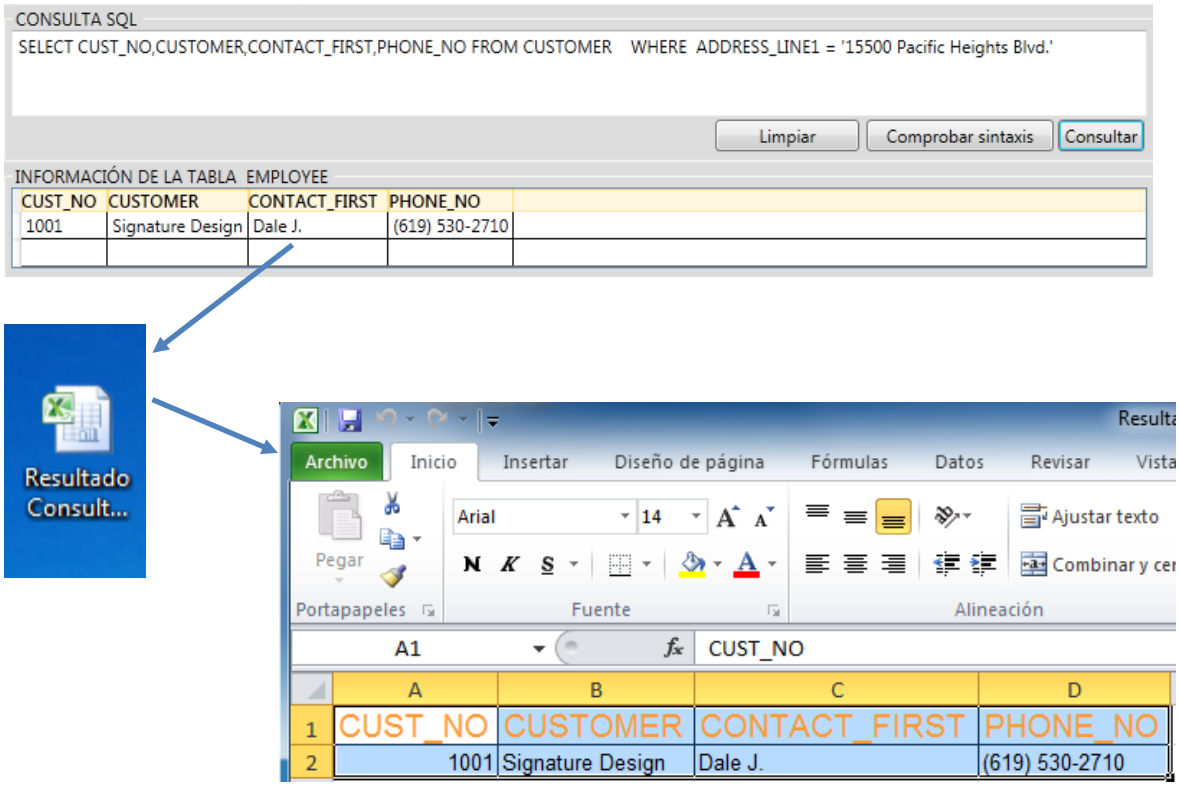
El resultado se verá manera formateada, aplicando los estilos definidos previamente en código XAML para diferencias correctamente los nombres de los campos y los valores que toman.



MU. Figura 9. Ejecución de sentencia SQL.

EXPORTAR RESULTADOS

Tras obtener un resultado, será posible migrarlos tanto a un archivo Excel como abrir un libro de Excel para verlos. Para ello habría que ir a la opción del menú Archivo/Exportar y seleccionar la opción correspondiente.

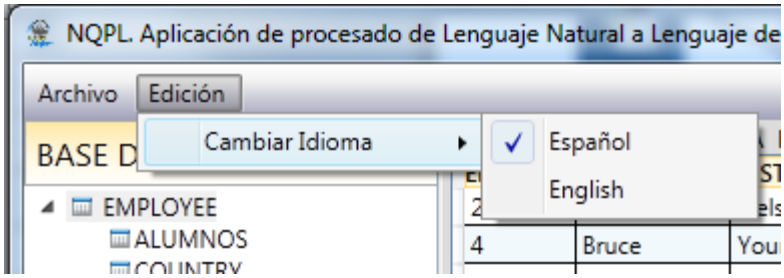


Mu. Figura 10. Exportar consulta a archivo Excel.

En cambio, exportar el resultado a una hoja de Excel lo que hace es mostrar directamente el resultado sin guardarlo en un fichero.

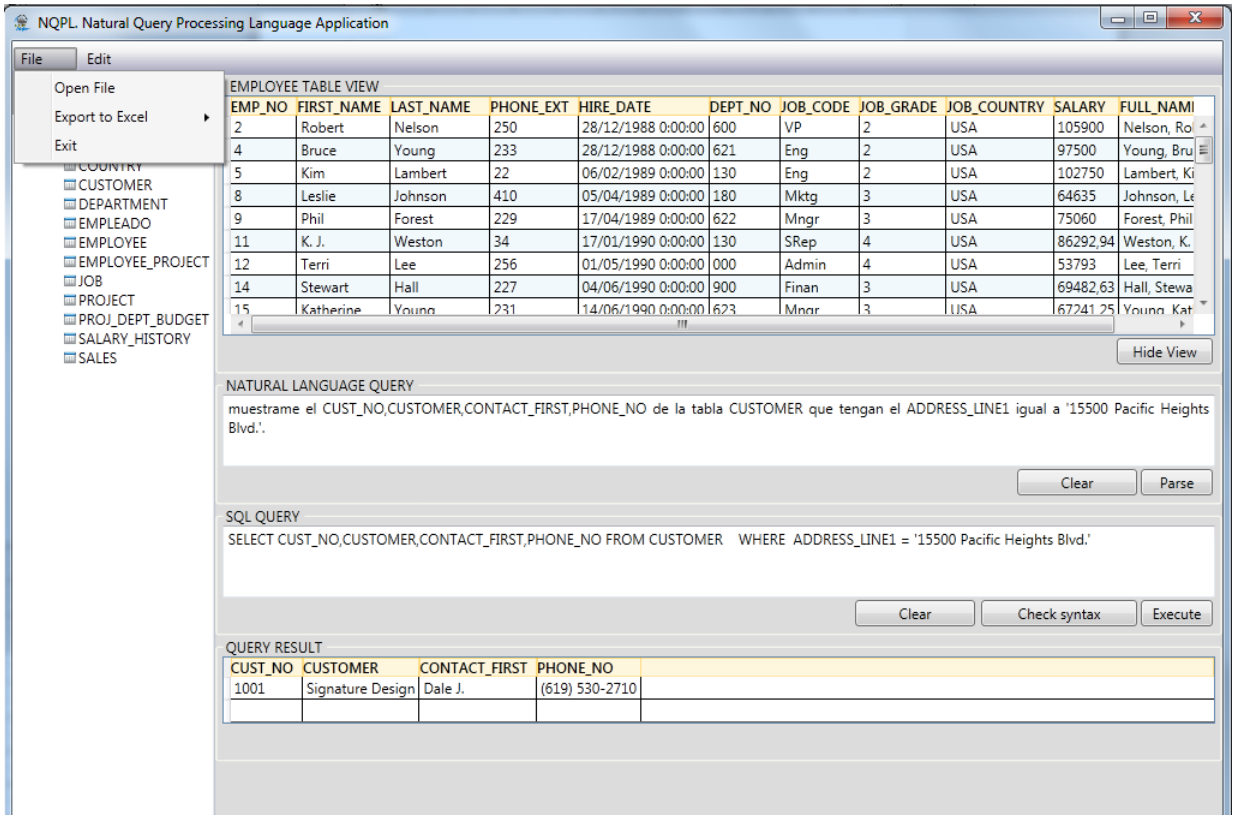
CAMBIAR EL IDIOMA DE LA INTERFAZ

La interfaz NQPL además, permite cambiar el idioma de visualización. Para ello hay que ir a la opción del menú Edición y seleccionar el lenguaje.



Mu. Figura 11. Cambio de idioma de la interfaz NQPL I.

El resultado del cambio de idioma de español a inglés tendría el siguiente aspecto, incluyendo mensajes emergentes:



Mu. Figura 12. Cambio de idioma de la interfaz NQPL II.

BIBLIOGRAFÍA

[1] *VSTO 3.0 for Office 2007 Programming (Vivek Thangaswamy).*

[2] *Paulo Reis, Joao Matias, Nuno Mamede. "A Natural Language Interface to Databases. A new dimension for an approach". 1997.*

[3] *C. J. Date. Introducción a los Sistemas de Bases de Datos. 7ª. Edición.*

[4] *"Aspectos de Calidad en el Desarrollo de Software Basado en Componentes", Manuel F. Bertoa, José M. Troya y Antonio Vallecillo, Dpto. Lenguajes y Ciencias de la Computación. Universidad de Málaga.*

[5] *G. Booch, J. Rumbaugh, I. Jacobson. "El Lenguaje Unificado de Modelado" .Addison Wesley Iberoamericana, 1999.*

[6] *Craig Larman. "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition).*

[7] *Kimmel Paul. "Manual de UML" (1ª Edición).*

[8] *Raúl Alarcón. "Diseño Orientado a Objetos con UML".*

[9] *Craig Larman. "UML y Patrones". Ed. Prentice Hall (1999).*

[10] *John Sharp. "Microsoft Visual Studio 2008. Step by Step."*

[11] *Adam Nathan. "Windows Presentation Foundation Unleashed"*

[12] *Carsten Thomsen. "Database Programming with C#" (2002).*

[13] *Jesse Liberty, Alex Horovitz. "Programming .NET 3.5". Ed. O'Reilly.*

[14] *AbrahamSilberchatz, Henry S. Korth, S. Sudarshan. "Fundamentos de Bases de Datos" .Ed. McGraw Hill (4ª Edición).*

[15] *Steve McConnell, "Desarrollo y Gestión de Proyectos Informáticos". Ed. McGraw Hill (1996).*